dpANS

Project T10/BSR INCITS 501

Revision 02
25 September 2015

Information technology – Security Features for SCSI Commands (SFSC)

T10 Technical Editor:

Ralph O. Weber
Western Digital Technologies
18484 Preston Road
Suite 102 PMB 178
Dallas, TX 75252
USA

Telephone: 214-912-1373
Email: Ralph dot Weber at WDC dot com

Reference number
ISO/IEC 14776-481 : 20xx
ANSI INCITS 501-20xx

Printed 9:07 AM Friday 25 September 2015

Points of Contact:

T10 Chair
Ralph Weber
Western Digital Technologies
18484 Preston Road, Suite 102, PMB 178
Dallas, TX 75252
Tel: 214-912-1373
Email: Ralph dot Weber at WDC dot com

T10 Vice-Chair
William Martin
Samsung Semiconductor Inc. (SSI)
7213 Marblethorpe Drive
Roseville, CA 95747-5925
Tel: 916-765-6875
Email: bill dot martin at ssi dot samsung dot com

INCITS Secretariat
INCITS Secretariat
1101 K Street NW, Suite 610
Washington, DC 20005-7031
Telephone: 202-737-8888
Facsimile: 202-638-4922
Email: incits@itic.org

T10 Web Site http://www.t10.org

T10 Reflector To subscribe, send e-mail to majordomo@T10.org with 'subscribe' in message body.
To unsubscribe, send e-mail to majordomo@T10.org with 'unsubscribe' in message body.
Internet address for distribution via T10 reflector: T10@T10.org

Purchase INCITS Standards
http://www.incits.org/standards-information/purchase-standards-or-download-dpans

ANSI (r)
INCITS 501-20xx

American National Standards for Information Systems -

Security Features for SCSI Commands (SFSC)

Secretariat
InterNational Committee for Information Technology Standards

Approved mm dd yy

American National Standards Institute, Inc.

Abstract

This standard defines security features for use by all SCSI devices. This standard defines the security mode that is basic to every device model and the parameter data that may apply to any device model.

American National Standard

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer. Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered and that effort be made toward their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he or she has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not confirming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give interpretation on any American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

CAUTION: The developers of this standard have requested that holders of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard.

As of the date of publication of this standard, following calls for the identification of patents that may be required for the implementation of the standard, notice of one or more claims has been received.

By publication of this standard, no position is taken with respect to the validity of this claim or of any rights in connection therewith. The known patent holder(s) has (have), however, filed a statement of willingness to grant a license under these rights on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such a license. Details may be obtained from the publisher.

No further patent search is conducted by the developer or the publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

Published by
American National Standards Institute
25 West 43rd Street 4th floor
New York, New York 10036-7422

Printed in the United States of America

Contents

Tables

Figures

Foreword

This foreword is not part of American National Standard INCITS 501-20xx.

This standard defines security features for use by all SCSI devices. This standard defines the security model that is basic to every device model and the parameter data that may apply to any device model.

With any technical document there may arise questions of interpretation as new products are implemented. INCITS has established procedures to issue technical opinions concerning the standards developed by INCITS. These procedures may result in SCSI Technical Information Bulletins being published by INCITS.

These Bulletins, while reflecting the opinion of the Technical Committee that developed the standard, are intended solely as supplementary information to other users of the standard. This standard, ANSI INCITS 501-20xx, as approved through the publication and voting procedures of the American National Standards Institute, is not altered by these bulletins. Any subsequent revision to this standard may or may not reflect the contents of these Technical Information Bulletins.

Current INCITS practice is to make standards and technical information bulletins available through:

http://www.incits.org/standards-information/purchase-standards-or-download-dpans

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent to the INCITS Secretariat, National Committee for Information Technology Standards, Information Technology Industry Council, 1101 K Street NW, Suite 610, Washington, DC 20005-7031.

This standard was processed and approved for submittal to ANSI by the InterNational Committee for Information Technology Standards (INCITS). Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time it approved this standard, INCITS had the following members:

AIM Global Inc

Adobe Systems Inc

American National Standards Institute

Apple

Department of Commerce - NIST

Distributed Management Task Force (DMTF)

EMC Corporation

Farance Inc

Futurewei Technologies Inc

GS1GO

Hewlett-Packard Company

IBM Corporation

IEEE

Intel Corporation

Microsoft Corporation

Oracle

Purdue University

Telecommunications Industry Association (TIA)

United States Dept of Defense

United States Dept of Homeland Security

Technical Committee T10 on SCSI Storage Interfaces, which developed and reviewed this standard, had the following members:

Ralph Weber, Chair
William Martin, Vice-Chair
John Geldman, Secretary

Organization Represented	*Name of Representative*
Amphenol Corporation	Gregory McSorley
	David Chan (Alt)
	Paul Coddington (Alt)
	Zhineng Fan (Alt)
	Adrian Green (Alt)
	Martin Li (Alt)
	Chris Lyon (Alt)
	Alex Persaud (Alt)
	Chansy Phommachanh (Alt)
	Michael Wingard (Alt)
	CN Wong (Alt)
	Matt Wright (Alt)
Avago Technologies	George Penokie
	Patrick Bashford (Alt)
	Brad Besmer (Alt)
	Srikiran Dravida (Alt)
	Rick Kutcipal (Alt)
	Bernhard Laschinsky (Alt)
	Harvey Newman (Alt)
	Robert Sheffield (Alt)
	Bill Voorhees (Alt)
Brocade	David Peterson
	Scott Kipp (Alt)
	Steven Wilson (Alt)
Dell, Inc.	Kevin Marks
	Mark Bokhan (Alt)
	Gary Kotzur (Alt)
	Bill Lynn (Alt)
	Ash McCarty (Alt)
	Daniel Oelke (Alt)
EMC Corporation	Gary Robinson
	David Black (Alt)
	Erin Bournival (Alt)
	George Ericson (Alt)
	Mickey Felton (Alt)
	Christopher Goonan (Alt)
	Marlon Ramroopsingh (Alt)

FCI Electronics	Donald Harper
	Brad Brubaker (Alt)
	Michael Craton (Alt)
	Michael Scholeno (Alt)
	Dave Sideck (Alt)
Foxconn Electronics	Fred Fons
	Gary Hsieh (Alt)
	Glenn Moore (Alt)
	Mike Shu (Alt)
	Miller Zhao (Alt)
Fujitsu America Inc	Kun Katsumata
	Osamu Kimura (Alt)
	Mark Malcolm (Alt)
	Gene Owens (Alt)
	Sandy Wilson (Alt)
Futurewei Technologies Inc	HengLiang Zhang
	Xiaoyu Ge (Alt)
	Xiaoyan He (Alt)
	Jia Shi (Alt)
Hewlett Packard Enterprise	Curtis Ballard
	Wayne Bellamy (Alt)
	Chris Cheng (Alt)
	Rob Elliott (Alt)
	Joe Foster (Alt)
	Barry Olawsky (Alt)
	Han Wang (Alt)
	Jeff Wolford (Alt)
HGST	Joe Breher
	David Brewer (Alt)
	Frank Chu (Alt)
	Jason Gao (Alt)
	Chet Mercado (Alt)
	Nadesan Narenthiran (Alt)
	Paul Suhler (Alt)
IBM Corporation	Kevin Butt
	Mike Osborne (Alt)
Intel Corporation	Pak-Lung Seto
	Richard Mellitz (Alt)
KnowledgeTek Inc	Dennis Moore
	Hugh Curley (Alt)
Marvell Semiconductor Inc	Paul Wassenberg
	Wei Liu (Alt)
	Wei Zhou (Alt)
Micron Technology Inc	John Geldman
	Jerry Barkley (Alt)
	Andrew Dunn (Alt)
	Neal Galbo (Alt)
	Michael George (Alt)
	Alan Haffner (Alt)
	Daniel Hubbard (Alt)
	Carl Mies (Alt)
	Niels Reimers (Alt)
	Bob Warren (Alt)

Microsoft Corporation	Lee Prewitt
	Douglas King (Alt)
	Paul Luber (Alt)
	Bryan Matthew (Alt)
	Steve Olsson (Alt)
Molex Inc	Jay Neer
	Cong Gao (Alt)
	Alex Haser (Alt)
	Ed Poh (Alt)
	Michael Rost (Alt)
	Darian Schulz (Alt)
	Scott Sommers (Alt)
NetApp Inc	Frederick Knight
	Chris Fore (Alt)
	Jaimon George (Alt)
	Ali Yavari (Alt)
OCZ Storage Solutions Inc.	Tom Friend
	Don Harwood (Alt)
Oracle	Dennis Appleyard
	Jon Allen (Alt)
	Seth Goldberg (Alt)
	Martin Petersen (Alt)
	Phi Tran (Alt)
	Lee Wan-Hui (Alt)
PMC-Sierra	Tim Symons
	David Allen (Alt)
	Adnan Jiwani (Alt)
	Keith Shaw (Alt)
	Gregory Tabor (Alt)
	Rod Zavari (Alt)
Quantum Corporation	Paul Stone
	Rod Wideman (Alt)
Samsung Semiconductor Inc (SSI)	William Martin
	Judy Brock (Alt)
	HeeChang Cho (Alt)
	KeunSoo Jo (Alt)
	Sreenivas Krishnan (Alt)
	Sung Lee (Alt)
	Bhavith M.P. (Alt)
	Truong Nguyen (Alt)
	Aishwarya Ravichandran (Alt)
SanDisk IL Ltd	Avraham Shimor
	Dave Landsman (Alt)
	Yoni Shternhell (Alt)
Seagate Technology	Gerald Houlder
	Michael Connolly (Alt)
	Alvin Cox (Alt)
	Neil Edmunds (Alt)
	Timothy Feldman (Alt)
	John Fleming (Alt)
	Jim Hatfield (Alt)
	Parag Maharana (Alt)
	Alan Westbury (Alt)
	Judy Westby (Alt)

TE Connectivity	Dan Gorenc
	Tom Grzysiewicz (Alt)
	John Hackman (Alt)
	Kyle Klinger (Alt)
	Joel Meyers (Alt)
	Andy Nowak (Alt)
	Eric Powell (Alt)
	Yasuo Sasaki (Alt)
	Scott Shuey (Alt)
	Robert Wertz (Alt)
Toshiba America Electronic Components Inc	Mike Fitzpatrick
	Dan Colegrove (Alt)
	Patrick Hery (Alt)
	Yuji Katori (Alt)
	Tom McGoldrick (Alt)
	Scott Wright (Alt)
VMware Inc	Patrick Dirks
	Deepak Babarjung (Alt)
	Neil H. MacLean (Alt)
	Mike Panas (Alt)
	Murali Rajagopal (Alt)
	Ahmad Tawil (Alt)
Western Digital Corporation	Curtis Stevens
	Marvin DeForest (Alt)
	Michael Koffman (Alt)
	Larry McMillan (Alt)
	Ralph Weber (Alt)

Introduction

The Security Features for SCSI Commands (SFSC) standard is divided into the following clauses and annexes:

Clause 1 is the scope.
Clause 2 enumerates the normative references that apply to this standard.
Clause 3 describes the definitions, symbols, and abbreviations used in this standard.
Clause 4 describes the security features model for all SCSI devices types.
Clause 5 defines security features parameters for use by all SCSI devices types.
Annex A describes security goals and threat model used by this standard. (informative)
Annex B identifies the differences between security protocols defined by other standards (e.g., IKEv2 (see RFC 7296)) and the equivalent protocols defined by this standard (e.g., the IKEv2-SCSI SA creation protocol). (informative)

The annexes provide information to assist with implementation of this standard. The information in the annexes applies to all command standards.

SCSI standards family

Figure 1 shows the relationship of this standard to the other standards and related projects in the SCSI family of standards as of the publication of this standard.

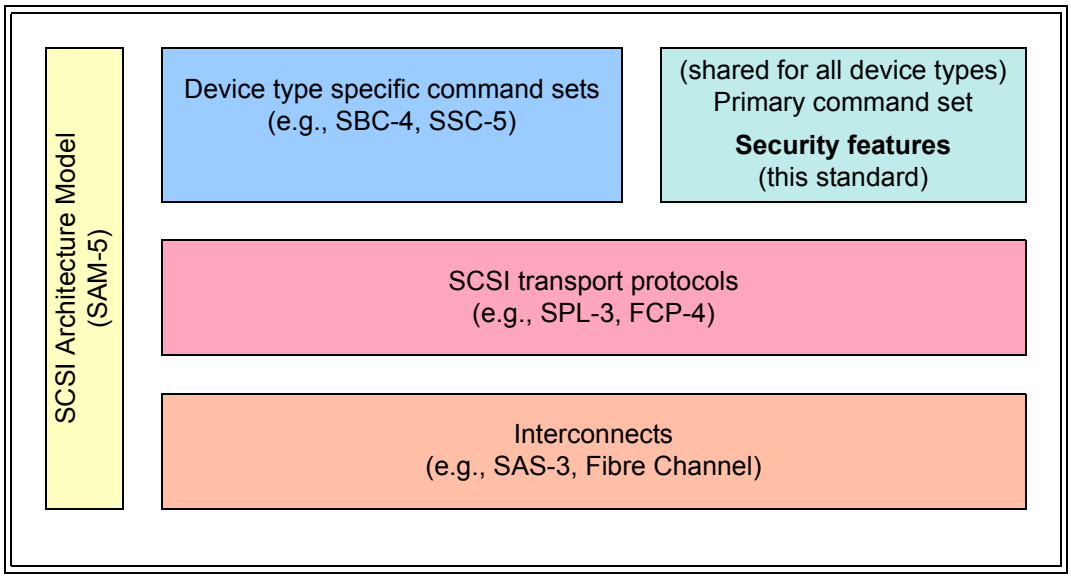

Figure 1 — SCSI document structure

The SCSI document structure in figure 1 is intended to show the general applicability of the documents to one another. Figure 1 is not intended to imply a relationship such as a hierarchy, protocol stack, or system architecture.

The term SCSI is used to refer to the family of standards described in this subclause.

AMERICAN NATIONAL STANDARD **INCITS 501-20xx**

American National Standard for Information Technology –

Security Features for SCSI Commands (SFSC)

1 Scope

The SCSI family of standards provides for many different types of SCSI devices (e.g., disks, tapes, media changers). This standard defines a device model that is applicable to all SCSI devices. Other command standards expand on the general SCSI device model in ways appropriate to specific types of SCSI devices.

The set of SCSI standards specifies the interfaces, functions, and operations necessary to ensure interoperability between conforming SCSI implementations. This standard is a functional description. Conforming implementations may employ any design technique that does not violate interoperability.

This standard defines security features for use by all SCSI devices. This standard defines the security model that is basic to every device model and the parameter data that may apply to any device model.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 14776-415, *SCSI Architecture Model - 5 (SAM-5)* (under national consideration)

ISO/IEC 14776-502, *SCSI Primary Commands - 5 (SPC-5)* (under national consideration)

ISO/IEC 14776-335, *SCSI Stream Commands - 5 (SSC-5)* (under national consideration)

ANSI/IEEE 1619.1-2007, *Standard for Authenticated Encryption with Length Expansion for Storage Devices*

ANSI INCITS 4-1986 (R2002), *Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII)*

ISO/IEC 646:1991, *Information technology - ISO 7-bit coded character set for information interchange (third edition)*

ISO/IEC 8859-1:1998, *Information technology - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1*

ISO/IEC 8859-2:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 2: Latin alphabet No. 2*

ISO/IEC 8859-3:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 3: Latin alphabet No. 3*

ISO/IEC 8859-4:1998, *Information technology - 8-bit single-byte coded graphic character sets - Part 4: Latin alphabet No. 4*

ISO/IEC 8859-5:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 5: Latin/Cyrillic alphabet*

ISO/IEC 8859-6:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 6: Latin/Arabic alphabet*

ISO/IEC 8859-7:1987, *Information processing - 8-bit single-byte coded graphic character sets - Part 7: Latin/Greek alphabet*

ISO/IEC 8859-8:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 8: Latin/Hebrew alphabet*

ISO/IEC 8859-9:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 9: Latin alphabet No. 5*

ISO/IEC 8859-10:1998, *Information technology - 8-bit single-byte coded graphic character sets - Part 10: Latin alphabet No. 6*

ISO/IEC 10646:2003, *Information technology - Universal Multiple-Octet Coded Character Set (UCS)*

ISO/IEC 9594-8:2001, *Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks [ITU-T Recommendation X.509:2000 – http://www.itu.int/ITU-T/]*

ISO/IEC 9594-8:2001/Cor 1:2002, *Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks*

ISO/IEC 9594-8:2001/Cor 2:2002, *Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks*

ISO/IEC 9594-8:2001/Cor 3:2005, *Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks*

RFC 2104, *HMAC: Keyed-Hashing for Message Authentication* [1]

RFC 2410, *The NULL Encryption Algorithm and Its Use With IPsec* [1]

RFC 3447, *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1* [1]

RFC 3526, *More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)* [1]

RFC 3566, *The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec* [1]

RFC 3602, *The AES-CBC Cipher Algorithm and Its Use with IPsec* [1]

RFC 3629, *UTF-8, a transformation format of ISO 10646* [1]

RFC 3766, *Determining Strengths For Public Keys Used For Exchanging Symmetric Keys* [1]

RFC 4086, *Randomness Requirements for Security* [1]

RFC 4106, *The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload* [1]

---

[1] Copies of the IETF RFCs may be obtained at http://www.ietf.org/.

RFC 4303, *IP Encapsulating Security Payload (ESP)* [1]

RFC 4309, *Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload* [1]

RFC 4434, *The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE)* [1]

RFC 4595, *Use of IKEv2 in the Fibre Channel Security Association Management Protocol* [1]

RFC 4754, *IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)* [1]

RFC 4868, *Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec* [1]

RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile* [1]

RFC 5755, *An Internet Attribute Certificate Profile for Authorization* [1]

RFC 5903, *Elliptic Curve Groups modulo a Prime (ECP Groups) for IKE and IKEv2* [1]

RFC 5996, *Internet Key Exchange Protocol Version 2 (IKEv2)* [1]

RFC 6151, *Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms* [1]

RFC 6818, *Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile* [1]

RFC 7296, *Internet Key Exchange Protocol Version 2 (IKEv2)* [1]

NIST SP (Special Publication) 800-38D, *Recommendation for Block Cipher Modes of Operation: Galois/Counter (GCM) Mode for Confidentiality and Authentication and GMAC* [2]

NIST SP 800-90 A revision 1, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators* (in development) [2]

FIPS 140-2, *Security Requirements for Cryptographic Modules* [2]

FIPS 140-3, *Security Requirements for Cryptographic Modules (in development)* [2]

FIPS 180-4, *Secure Hash Standard* [2]

FIPS 198-1, *The Keyed-Hash Message Authentication Code (HMAC)* [2]

[2] Copies of the NIST and FIPS standards may be obtained at http://csrc.nist.gov/publications/fips/index.html.

3 Definitions, symbols, abbreviations, and conventions

3.1 Definitions

3.1.1 additional authenticated data (AAD)
bytes of data input to a combined mode encryption algorithm (e.g., AES-GCM) as part of integrity checking computations (e.g., the IKEv2-SCSI Encrypted payload (see 5.3.5.11) uses AAD)

3.1.2 additional sense code
combination of the ADDITIONAL SENSE CODE field and the ADDITIONAL SENSE CODE QUALIFIER field in sense data (see SPC-5)

3.1.3 application client
object that is the source of SCSI commands

Note 1 to entry: See SAM-5.

3.1.4 byte
sequence of eight contiguous bits considered as a unit

3.1.5 command
request describing a unit of work to be performed by a device server

Note 1 to entry: See SAM-5.

3.1.6 command descriptor block (CDB)
structure used to communicate commands from an application client to a device server (see SPC-5)

Note 1 to entry: A CDB may have a fixed length of up to 16 bytes or a variable length of between 12 and 260 bytes.

3.1.7 command standard
SCSI standard that defines the model, commands, and parameter data for a device type (e.g., SBC-3, SSC-4, SMC-3, MMC-6, or SES-2)

3.1.8 company_id
synonym for OUI (see 3.1.47)

3.1.9 cryptographic command sequence (CCS)
defined sequence of SECURITY PROTOCOL IN commands (see SPC-5) and SECURITY PROTOCOL OUT commands (see SPC-5) that realize the cryptographic protocol of a specified security operation (e.g., the SECURITY PROTOCOL IN commands and SECURITY PROTOCOL OUT commands in the Key Exchange step and Authentication step, if any, of an IKEv2-SCSI (see 3.1.27) SA creation transaction (see 3.1.59))

3.1.10 Data-In Buffer
buffer specified by the application client to receive data from the device server during the processing of a command

Note 1 to entry: See SAM-5.

3.1.11 Data-Out Buffer
buffer specified by the application client to supply data that is sent from the application client to the device server during the processing of a command

Note 1 to entry: See SAM-5.

3.1.12 deferred error
CHECK CONDITION status and sense data that is returned as the result of an error or exception condition that occurred during processing of a previous command for which GOOD status or CONDITION MET status has already been returned

Note 1 to entry: See SPC-5.

3.1.13 designation
distinguishing name, identifier, or title

3.1.14 device server
object within a logical unit that processes SCSI commands according to the rules of command management

Note 1 to entry: See SAM-5.

3.1.15 device type
device model implemented by the logical unit and indicated to the application client by the contents of the PERIPHERAL DEVICE TYPE field in the standard INQUIRY data (see SPC-5)

3.1.16 Encapsulating Security Payload for SCSI (ESP-SCSI)
method for transferring encrypted and/or integrity checked parameter data in Data-In Buffers and/or Data-Out Buffers based on Encapsulating Security Payload (see RFC 4303)

Note 1 to entry: See 4.1.5.

3.1.17 Extended CDB (XCDB)
CDB that contains another CDB and additional information to support extra processing (e.g., security features)

Note 1 to entry: See SPC-5.

3.1.18 Extended Unique Identifier, a 48-bit globally unique identifier (EUI-48)
48-bit identifier that is globally unique

Note 1 to entry: The IEEE maintains a tutorial describing EUI-48 at http://standards.ieee.org/regauth/oui/tutorials/EUI48.html.

3.1.19 Extended Unique Identifier, a 64-bit globally unique identifier (EUI-64)
64-bit identifier that is globally unique

Note 1 to entry: The IEEE maintains a tutorial describing EUI-64 at http://standards.ieee.org/regauth/oui/tutorials/EUI64.html.

3.1.20 field
group of one or more contiguous bits, a part of a larger structure such as a CDB (see 3.1.6) or sense data (see 3.1.78)

3.1.21 hard reset
condition resulting from the events defined by SAM-5 in which the SCSI device performs the hard reset operations described in SAM-5, this standard, and the applicable command standards

3.1.22 hashed message authentication code (HMAC)
type of message authentication code that is calculated using the cryptographic hash function as defined in FIPS 198-1 (see clause 2) in combination with a secret key

3.1.23 I_T nexus
nexus between a SCSI initiator port and a SCSI target port

Note 1 to entry: See SAM-5.

3.1.24 I_T nexus loss
condition resulting from the events defined by SAM-5

Note 1 to entry: In response to this condition, the SCSI device performs the I_T nexus loss operations described in SAM-5, this standard, and other applicable standards.

3.1.25 I_T_L nexus
nexus between a SCSI initiator port, a SCSI target port, and a logical unit

Note 1 to entry: See SAM-5.

3.1.26 IEEE company_id
synonym for OUI (see 3.1.47)

3.1.27 IKEv2-SCSI
Internet Key Exchange protocol version 2 for SCSI

Note 1 to entry: See 4.1.3.

3.1.28 IKEv2-SCSI keys
shared keys (see 3.1.81) used to provide security for IKEv2-SCSI operations (e.g., creation and deletion of the SA)

Note 1 to entry: IKEv2-SCSI keys used after SA creation is complete are maintained in the MGMT_DATA SA parameter (see 3.1.62).

Note 2 to entry: The shared key names used by IKEv2-SCSI are listed in 4.1.3.4.

3.1.29 IKEv2-SCSI CCS
CCS (see 3.1.9) that is the Key Exchange step and Authentication step, if any, of an IKEv2-SCSI (see 3.1.27) SA creation transaction (see 3.1.59)

3.1.30 information unit (IU)
delimited and sequenced set of information in a format appropriate for transport by the service delivery subsystem (e.g., a CDB for a specific SCSI command)

3.1.31 initiator port
synonymous with SCSI initiator port (see 3.1.69)

3.1.32 initiator port name
name (see 3.1.42) of a SCSI initiator port

3.1.33 integrity check value
value used to cryptographically validate the integrity of a specified set of bytes that contain specified data based on a shared key (see 3.1.81)

3.1.34 internet assigned numbers authority (IANA)
service that assigns and manages registries of code values (i.e., code points) for the IETF (see http://www.ietf.org) and other users of the public internet (see http://www.iana.org)

Note 1 to entry: IANA maintains code values used by Internet Key Exchange version 2 (IKEv2) at http://www.iana.org/assignments/ikev2-parameters

3.1.35 key derivation function (KDF)
algorithm that is used to derive cryptographic keying material from a shared secret and other information

3.1.36 least significant bit (LSB)
in a binary code, the bit or bit position with the smallest numerical weighting in a group of bits that, when taken as a whole, represent a numerical value

EXAMPLE – In the number 0001b, the bit that is set to one.

3.1.37 left-aligned
type of field containing ASCII data in which unused bytes are placed at the end of the field (i.e., highest offset) and are filled with ASCII space (20h) characters

Note 1 to entry: See SPC-5.

3.1.38 logical unit
externally addressable entity within a SCSI target device that implements a SCSI device model and contains a device server

Note 1 to entry: See SAM-5.

3.1.39 logical unit number (LUN)
encoded 64-bit identifier for a logical unit

Note 1 to entry: See SAM-5.

3.1.40 logical unit reset
condition resulting from the events defined by SAM-5

Note 1 to entry: In response to this condition, the logical unit performs the logical unit reset operations described in SAM-5, this standard, and other applicable standards.

3.1.41 most significant bit (MSB)
in a binary code, the bit or bit position with the largest numerical weighting in a group of bits that, when taken as a whole, represent a numerical value (e.g., in the number 1000b, the bit that is set to one)

3.1.42 name
label of an object that is unique within a specified context and should never change (e.g., the term name and world wide identifier (WWID) may be interchangeable)

3.1.43 nexus
relationship between two SCSI devices, and the SCSI initiator port and SCSI target port objects within those SCSI devices

Note 1 to entry: See SAM-5.

3.1.44 nonce
value that is used one and only one time to provide uniqueness to a value (e.g., a secure cryptographic key) in whose derivation the nonce participates or to uniquely identify a single instance of something (e.g., a timestamp) exchanged between an application client and a device server

3.1.45 null-padded
type of field in which unused bytes are placed at the end of the field (i.e., highest offset) and are filled with ASCII null (00h) characters

Note 1 to entry: See SPC-5.

3.1.46 null-terminated
type of field in which the last used byte (i.e., highest offset) is required to contain an ASCII null (00h) character

Note 1 to entry: See SPC-5.

3.1.47 organizationally unique identifier (OUI)
numeric identifier that is assigned by the IEEE such that no assigned identifiers are identical

Note 1 to entry: OUI is equivalent to company_id or IEEE company_id.

Note 2 to entry: The IEEE prefers OUI for EUI-48 identifiers (see 3.1.18) and company_id for EUI-64 identifiers (see 3.1.19). However, the numeric identifier is called an OUI when that identifier is assigned by the IEEE.

Note 3 to entry: The IEEE maintains a tutorial describing the OUI at http://standards.ieee.org/regauth/oui/.

3.1.48 page
regular parameter structure or format used by several commands and identified with a value known as a page code

3.1.49 peripheral device
object that connects to a logical unit

3.1.50 peripheral device type
synonym for device type (see 3.1.15)

3.1.51 plaintext
unencrypted bytes

3.1.52 power cycle
power being removed from and later applied to a SCSI device

3.1.53 pre-shared key
shared key (see 3.1.81) that is established by a method outside the scope of this standard prior to the initiation of the function that requires its use

Note 1 to entry: The requirements on pre-shared keys that are particular to this standard are described in 4.1.3.3.2.

Note 2 to entry: Some of the RFCs referenced by this standard use the name pre-shared secret for the same information that this standard calls a pre-shared key.

Note 3 to entry: See SPC-5.

3.1.54 protocol specific
requirement that is defined by a SCSI transport protocol standard (see 3.1.73)

Note 1 to entry: See SAM-5.

3.1.55 protocol standard
synonymous with SCSI transport protocol standard (see 3.1.73)

3.1.56 random nonce
secure random number (see 4.2) that has a negligible chance of repeating and whose uses include providing significant uniqueness and randomness in cryptographic calculations

3.1.57 request for comment (RFC)
name given to standards developed by the Internet Engineering Task Force

Note 1 to entry: See http://www.ietf.org/.

3.1.58 right-aligned
type of field containing ASCII data in which unused bytes are placed at the start of the field (i.e., lowest offset) and are filled with ASCII space (20h) characters

Note 1 to entry: See SPC-5.

3.1.59 SA creation transaction
any sequence of SECURITY PROTOCOL IN commands (see SPC-5) and SECURITY PROTOCOL OUT commands (see SPC-5) that is used to create an SA between an application client and device server

Note 1 to entry: A CCS (see 3.1.9) is a kind of SA creation transaction.

3.1.60 SA generation
computation and initialization of the SA parameter values (see 3.1.62) required to create an SA (see 3.1.76)

Note 1 to entry: SA generation is performed separately by the application client and device server after all SCSI commands required to create an SA have been performed without error

Note 2 to entry: See 4.1.3.9.

3.1.61 SA keys
shared keys (see 3.1.81) that are maintained in the USAGE_DATA SA parameter (see 3.1.62)

Note 1 to entry: SA keys are used to provide security for the operations that use the SA (e.g., encryption of command parameter data).

Note 2 to entry: The shared key names used by IKEv2-SCSI are listed in 4.1.3.4.

3.1.62 SA parameters
parameters stored by both an application client and a device server

Note 1 to entry: SA parameters are associated with one SA (see 3.1.76) and identified by a pair of SAIs (see 3.1.77).

Note 2 to entry: See 4.1.1.2.

3.1.63 SA participant
application client or device server that participates in the creation or use of an SA (see 3.1.76)

3.1.64 salt bytes
sequence of bytes whose contents are unpredictable in advance of their use

Note 1 to entry: Salt bytes are kept secret prior to their use and disclosed during their use (e.g., bytes added to a cryptographic operation to increase the entropy in the result (see RFC 4106)).

3.1.65 SCSI device
device that contains one or more SCSI ports that are each connected to a service delivery subsystem and supports a SCSI application protocol

Note 1 to entry: See SAM-5.

3.1.66 SCSI device name
name (see 3.1.42) of a SCSI device that is world wide unique within the protocol of a SCSI domain (see 3.1.67) in which the SCSI device has SCSI ports (see 3.1.70)

Note 1 to entry: The SCSI device name may be made available to other SCSI devices or SCSI ports in protocol specific ways.

Note 2 to entry: See SAM-5.

3.1.67 SCSI domain
interconnection of two or more SCSI devices and a service delivery subsystem

3.1.68 SCSI initiator device
SCSI device (see 3.1.65) containing application clients (see 3.1.3) and SCSI initiator ports (see 3.1.69)

Note 1 to entry: SCSI initiator devices originate device service and task management requests (see SAM-5) to be processed by a SCSI target device (see 3.1.71) and receive device service and task management responses from SCSI target devices.

Note 2 to entry: See SAM-5.

3.1.69 SCSI initiator port
SCSI initiator device (see SAM-5) object that acts as the connection between application clients and a service delivery subsystem through which requests and responses are routed

Note 1 to entry: See SAM-5.

3.1.70 SCSI port
SCSI initiator port (see 3.1.69) or SCSI target port (see 3.1.72)

3.1.71 SCSI target device
SCSI device (see 3.1.65) containing logical units (see 3.1.38) and SCSI target ports (see 3.1.72)

Note 1 to entry: SCSI target devices receive device service and task management requests (see SAM-5) for processing and send device service and task management responses to SCSI initiator devices.

Note 2 to entry: See SAM-5.

3.1.72 SCSI target port
SCSI target device (see SAM-5) object that acts as the connection between device servers and task managers and a service delivery subsystem through which requests and responses are routed

Note 1 to entry: See SAM-5.

3.1.73 SCSI transport protocol standard
SCSI standard that defines a SCSI transport protocol (e.g., FCP-4, SPL-3, SRP, or SBP-3)

3.1.74 secure hash algorithm (SHA)
secure hash algorithm (e.g., SHA-1) defined in FIPS 180-4, Secure Hash Standard (SHS)

Note 1 to entry: See clause 2.

3.1.75 secure random number
random number that is generated in ways that protect the random number from security attacks

Note 1 to entry: See 4.2.

3.1.76 security association (SA)
relationship and associated security processing between an application client and a device server that is used to apply security functions (e.g., integrity checking, encryption) to data that is transferred in either direction

Note 1 to entry: See 4.1.1.

3.1.77 security association index (SAI)
number representing the parameters for an SA as stored internally by the application client or device server

Note 1 to entry: In other security models, this value is called the security parameters index (SPI).

Note 2 to entry: See 4.1.1.

3.1.78 sense data
data describing an error, exceptional condition, or completion information

Note 1 to entry: See SPC-5.

3.1.79 sense key
contents of the SENSE KEY field in sense data (see SPC-5)

3.1.80 service delivery subsystem
that part of a SCSI domain that transmits service requests to a logical unit or SCSI target device and returns logical unit or SCSI target device responses to a SCSI initiator device

Note 1 to entry: See SAM-5.

3.1.81 shared key (SK)
cryptographically protected secret (e.g., with more randomness (see RFC 4089) than a password) that is known only to a defined and limited set of entities (e.g., one application client and one device server)

Note 1 to entry: The shared key names used by IKEv2-SCSI are listed in 4.1.3.4.

3.1.82 SK_ai
shared key (see 3.1.81) used for integrity checking data in a Data-Out Buffer (e.g., integrity checking the Encrypted payload in an IKEv2-SCSI Authentication step SECURITY PROTOCOL OUT command (see 4.1.3.7.2))

3.1.83 SK_ar
shared key (see 3.1.81) used for integrity checking data in a Data-In Buffer (e.g., integrity checking the Encrypted payload in an IKEv2-SCSI Authentication step SECURITY PROTOCOL IN command (see 4.1.3.7.3))

3.1.84 SK_d
shared key (see 3.1.81) KDF input material use to generate the shared keys stored in the KEYMAT SA Parameter (see 3.1.62)

3.1.85 SK_ei
shared key (see 3.1.81) used for encrypting data in a Data-Out Buffer (e.g., encrypting the Encrypted payload in an IKEv2-SCSI Authentication step SECURITY PROTOCOL OUT command (see 4.1.3.7.2))

3.1.86 SK_er
shared key (see 3.1.81) used for encrypting data in a Data-In Buffer (e.g., encrypting the Encrypted payload in an IKEv2-SCSI Authentication step SECURITY PROTOCOL IN command (see 4.1.3.7.3))

3.1.87 SK_pi
shared key (see 3.1.81) used to construct the IKEv2-SCSI Data-Out Buffer Authentication payload (see 4.1.3.7.2)

3.1.88 SK_pr
shared key (see 3.1.81) used to construct the IKEv2-SCSI Data-In Buffer Authentication payload (see 4.1.3.7.3)

3.1.89 status
one byte of response information that contains a coded value defined in SAM-5, transferred from a device server to an application client upon completion of each command

Note 1 to entry: See SAM-5.

3.1.90 system
one or more SCSI domains operating as a single configuration

3.1.91 target port
synonymous with SCSI target port (see 3.1.72)

3.1.92 target port name
name (see 3.1.42) of a SCSI target port

3.1.93 task set
group of commands within a logical unit, whose interaction is dependent on the task management (i.e., queuing) and ACA rules

Note 1 to entry: The Control mode page (see SPC-5) defines management capabilities for task sets.

Note 2 to entry: See SAM-5.

3.1.94 trust anchor
authoritative entity for which trust is assumed and not derived

3.1.95 unit attention condition
asynchronous status information that a logical unit establishes to report to the initiator ports associated with one or more I_T nexuses

Note 1 to entry: See SAM-5.

3.1.96 user data
data contained in logical blocks (e.g., see SBC-3) that is not protection information (see SPC-5)

Note 1 to entry: Each command standard (see 3.1.7) may provide its own definition of user data.

3.1.97 UTF-8
character set that is a transformation format of the character set defined by ISO 10646

Note 1 to entry: See RFC 3629.

3.1.98 word
sequence of 16 contiguous bits considered as a unit

3.1.99 zero-padded
type of field in which unused bytes are placed at the end of the field (i.e., highest offset) and are filled with zeros

Note 1 to entry: See SPC-5.

3.2 Abbreviations and symbols

3.2.1 Abbreviations

See Clause 2 for abbreviations of standards bodies (e.g., ISO). Abbreviations used in this standard:

Abbreviation	Meaning
AAD	Additional Authenticated Data (see 3.1.1)
ACA	Auto Contingent Allegiance (see SAM-5)
AES	Advanced Encryption Standards
AES-GCM	Advanced Encryption Standard - Galois Counter Mode (see RFC 4106)
ASCII	American Standard Code for Information Interchange (see clause 2)
ASN.1	Abstract Syntax Notation One (see clause 2)
CCS	Cryptographic Command Sequence (see 3.1.9)
CDB	Command Descriptor Block (see 3.1.6)
CSEC	Communications Security Establishment Canada
ECP	Elliptic Curve group modulo Prime (see RFC 5903)
ECDSA	Elliptic Curve Digital Signature Algorithm (see RFC 4754)
ESP-SCSI	Encapsulating Security Payload for SCSI (see 3.1.16)
EUI-48	Extended Unique Identifier, a 48-bit globally unique identifier (see 3.1.18)
EUI-64	Extended Unique Identifier, a 64-bit globally unique identifier (see 3.1.19)
FC-SP-2	Fibre Channel Security Protocols-2
FIPS	Federal Information Processing Standard
HMAC	Hashed Message Authentication Code (see 3.1.22)
HTTP	Hypertext Transfer Protocol (see RFC 2616)
IANA	Internet Assigned Numbers Authority (see 3.1.34)
ID	Identifier or Identification
IEEE	Institute of Electrical and Electronics Engineers
IKEv2	Internet Key Exchange version 2 (see clause 2)
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IU	Information Unit
KDF	Key Derivation Function (see 3.1.35)
LSB	Least Significant Bit (see 3.1.36)
LUN	Logical Unit Number (see 3.1.39)
MAC	Message Authentication Code
MODP	Modular exponential (see RFC 3526)
MSB	Most Significant Bit (see 3.1.41)
OUI	Organizationally Unique Identifier (see 3.1.47)
PKI	Public Key Infrastructure (see RFC 5280 and RFC 6818)
PRF	Pseudo-Random Function (see RFC 7296)
RFC	Request For Comments (see 3.1.57)
SA	Security Association (see 3.1.76)
SAI	Security Association Index (see 3.1.77)
SAM-5	SCSI Architecture Model - 5
SAUT	SA Usage Type (see 5.3.5.14)
SBC-4	SCSI Block Commands - 4
SCSI	Small Computer System Interface
SFSC	Security Features for SCSI Commands (this standard)
SHA-1	Secure Hash Algorithm, 160 bits (see 3.1.74)

Abbreviation	Meaning
SHA-256	Secure Hash Algorithm, 256 bits (see 3.1.74)
SHA-384	Secure Hash Algorithm, 384 bits (see 3.1.74)
SHA-512	Secure Hash Algorithm, 512 bits (see 3.1.74)
SK	Shared Key (see 3.1.81)
SPC-5	SCSI Primary Commands - 5
SSC-5	SCSI Stream Commands - 5
URI	Uniform Resource Identifier (see RFC 3986)
URL	Uniform Resource Locator (see RFC 3986)
VPD	Vital Product Data (see SPC-5)
VS	Vendor Specific (see 3.3.12)
XCDB	Extended CDB (see 3.1.17)

3.2.2 Symbols

Symbols used in this standard:

Symbol	Meaning
||	concatenation
n/a	not applicable
x, xx	any valid value for a bit or field

3.2.3 Mathematical operators

Mathematical operators used in this standard:

Mathematical operator	Meaning
+	plus
–	minus
/	divided by
×	multiplied by
<	less than
≤	less than or equal to
>	greater than
≥	greater than or equal to
=	equal to
≠	not equal to

3.3 Keywords

3.3.1 invalid

keyword used to describe an illegal or unsupported bit, byte, word, field or code value

Note 1 to entry: Receipt by the device server of an invalid bit, byte, word, field or code value shall be reported as an error.

3.3.2 mandatory
keyword indicating an item that is required to be implemented as defined in this standard

3.3.3 may
keyword indicating flexibility of choice with no implied preference

3.3.4 may not
keyword indicating flexibility of choice with no implied preference

3.3.5 obsolete
keyword indicating that an item was defined in prior SCSI standards but has been removed from this standard

3.3.6 option, optional
keywords that describe features that are not required to be implemented by this standard

Note 1 to entry: If any optional feature defined by this standard is implemented, then it shall be implemented as defined in this standard.

3.3.7 prohibited
keyword used to describe a feature, function, or coded value that is defined in a non-SCSI standard (i.e., a standard that is not a member of the SCSI family of standards) to which this standard makes a normative reference where the use of said feature, function, or coded value is not allowed for implementations of this standard

3.3.8 reserved
keyword referring to bits, bytes, words, fields and code values that are set aside for future standardization

Note 1 to entry: A reserved bit, byte, word or field shall be set to zero, or in accordance with a future extension to this standard.

Note 2 to entry: Recipients are not required to check reserved bits, bytes, words or fields for zero values.

Note 3 to entry: Receipt of reserved code values in defined fields shall be reported as an error.

3.3.9 restricted
keyword referring to bits, bytes, words, and fields that are set aside for other identified standardization purposes

Note 1 to entry: A restricted bit, byte, word, or field shall be treated as a reserved bit, byte, word or field in the context where the restricted designation appears.

3.3.10 shall
keyword indicating a mandatory requirement

Note 1 to entry: Designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard.

3.3.11 should
keyword indicating flexibility of choice with a strongly preferred alternative

3.3.12 vendor specific (VS)
something (e.g., a bit, field, code value) that is not defined by this standard

Note 1 to entry: Specification of the referenced item is determined by the SCSI device vendor and may be used differently in various implementations.

3.4 Conventions

Certain words and terms used in this standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in 3.1 or in the text where they first appear. Names of commands, statuses, sense keys, and additional sense codes are in all uppercase (e.g., REQUEST SENSE). Lowercase is used for words having the normal English meaning.

If there is more than one CDB length for a particular command (e.g., MODE SENSE(6) and MODE SENSE(10)) and the name of the command is used in a sentence without any CDB length descriptor (e.g., MODE SENSE), then the condition described in the sentence applies to all CDB lengths for that command.

The names of fields are in small uppercase (e.g., ALLOCATION LENGTH). When a field name is a concatenation of acronyms, uppercase letters may be used for readability (e.g., NORMACA). Normal case is used when the contents of a field are being discussed. Fields containing only one bit are usually referred to as the name bit instead of the name field.

When the value of the bit or field is not relevant, x or xx appears in place of a specific value.

Lists sequenced by lowercase or uppercase letters show no ordering relationship between the listed items.

EXAMPLE 1 – The following list shows no relationship between the colors named:

a) red, specificity one of the following colors:
   A) crimson; or
   B) amber;
b) blue; or
c) green.

Lists sequenced by numbers show an ordering relationship between the listed items.

EXAMPLE 2 – The following list shows the order in which a page is meant to be read:

1) top;
2) middle; and
3) bottom.

Lists are associated with an introductory paragraph or phrase, and are numbered relative to that paragraph or phrase (i.e., all lists begin with an a) or 1) entry).

If a conflict arises between text, tables, or figures, the order of precedence to resolve the conflicts is text; then tables; and finally figures. Not all tables or figures are fully described in the text. Tables show data format and values. Notes do not constitute any requirements for implementors.

3.5 Numeric and character conventions

3.5.1 Numeric conventions

A binary number is represented in this standard by any sequence of digits comprised of only the Arabic numerals 0 and 1 immediately followed by a lower-case b (e.g., 0101b). Underscores or spaces may be included in binary number representations to increase readability or delineate field boundaries (e.g., 0 0101 1010b or 0_0101_1010b).

A hexadecimal number is represented in this standard by any sequence of digits comprised of only the Arabic numerals 0 to 9 and/or the upper-case English letters A to F immediately followed by a lower-case h (e.g., FA23h). Underscores or spaces may be included in hexadecimal number representations to increase readability or delineate field boundaries (e.g., B FD8C FA23h or B_FD8C_FA23h).

A decimal number is represented in this standard by any sequence of digits comprised of only the Arabic numerals 0 to 9 not immediately followed by a lower-case b or lower-case h (e.g., 25).

A range of numeric values is represented in this standard in the form “a to z”, where a is the first value included in the range, all values between a and z are included in the range, and z is the last value included in the range (e.g., the representation “0h to 3h” includes the values 0h, 1h, 2h, and 3h).

This standard uses the following conventions for representing decimal numbers:

a) the decimal separator (i.e., separating the integer and fractional portions of the number) is a period;
b) the thousands separator (i.e., separating groups of three digits in a portion of the number) is a space;
c) the thousands separator is used in both the integer portion and the fraction portion of a number; and
d) the decimal representation for a year is 1999 not 1 999.

Table 1 shows some examples of decimal numbers represented using various conventions.

Table 1 — Numbering conventions examples

French	English	This standard
0,6	0.6	0.6
3,141 592 65	3.14159265	3.141 592 65
1 000	1,000	1 000
1 323 462,95	1,323,462.95	1 323 462.95

A decimal number represented in this standard with an overline over one or more digits following the decimal point is a number where the overlined digits are infinitely repeating (e.g., $666.\overline{6}$ means 666.666 666… or 666 2/3 and $12.\overline{142\ 857}$ means 12.142 857 142 857… or 12 1/7).

3.5.2 Units of measure

This standard represents values using both decimal units of measure and binary units of measure. Values are represented by the following formats:

a) for values based on decimal units of measure:
    1) numerical value (e.g., 100);
    2) space;
    3) prefix symbol and unit:
        1) decimal prefix symbol (e.g., M) (see table 2); and
        2) unit abbreviation (e.g., B);

    and
b) for values based on binary units of measure:
    1) numerical value (e.g., 1 024);
    2) space;
    3) prefix symbol and unit:
        1) binary prefix symbol (e.g., Gi) (see table 2); and
        2) unit abbreviation (e.g., b).

Table 2 compares the prefix, symbols, and power of the binary and decimal units.

Table 2 — Comparison of decimal prefixes and binary prefixes

Decimal			Binary		
Prefix name	**Prefix symbol**	**Power (base-10)**	**Prefix name**	**Prefix symbol**	**Power (base-2)**
kilo	k	10^3	kibi	Ki	2^{10}
mega	M	10^6	mebi	Mi	2^{20}
giga	G	10^9	gibi	Gi	2^{30}
tera	T	10^{12}	tebi	Ti	2^{40}
peta	P	10^{15}	pebi	Pi	2^{50}
exa	E	10^{18}	exbi	Ei	2^{60}
zetta	Z	10^{21}	zebi	Zi	2^{70}
yotta	Y	10^{24}	yobi	Yi	2^{80}

3.5.3 Byte encoded character strings conventions

When this standard requires one or more bytes to contain specific encoded characters, the specific characters are enclosed in single quotation marks. The single quotation marks identify the start and end of the characters that are required to be encoded but are not themselves to be encoded. The characters that are to be encoded are shown in the case that is to be encoded.

An ASCII space character (i.e., 20h) may be represented in a string by the character '¬' (e.g., 'SCSI¬device').

The encoded characters and the single quotation marks that enclose them are preceded by text that specifies the character encoding methodology and the number of characters required to be encoded.

EXAMPLE - Using the notation described in this subclause, stating that the eleven ASCII characters 'SCSI device' represent encoded characters is the same as writing out the following sequence of byte values: 53h 43h 53h 49h 20h 64h 65h 76h 69h 63h 65h.

3.6 Bit and byte ordering

This subclause describes the representation of fields in a table that defines the format of a SCSI structure (e.g., the format of a CDB).

If a field consists of more than one bit and contains a single value (e.g., a number), the LSB is shown on the right and the MSB is shown on the left (e.g., in a byte, bit 7 is the MSB and is shown on the left, and bit 0 is the LSB and is shown on the right). The MSB and LSB are not labeled if the field consists of 8 or fewer bits.

If a field consists of more than one byte and contains a single value, the byte containing the MSB is stored at the lowest address and the byte containing the LSB is stored at the highest address (i.e., big-endian byte ordering). The MSB and LSB are labeled.

If a field consists of more than one byte and always contains multiple fields each with their own values (e.g., a descriptor), then there is no MSB and LSB of the field itself and thus there are no MSB and LSB labels. Each

individual field has an MSB and LSB that are labeled as appropriate in the table, if any, that describes the format of the sub-structure having multiple fields.

If a field that consists of more than one byte contains either multiple fields each with their own values or a single value, then:

a) the MSB and LSB are labeled and they apply as described in this subclause whenever the field contains a single value; and
b) the labels on constituent fields supersede the single value labels whenever the field contains multiple fields.

If a field contains a text string (e.g., ASCII or UTF-8), the MSB label is the MSB of the first character and the LSB label is the LSB of the last character.

When required for clarity, multiple byte fields may be represented with only two rows in a table. This condition is represented by values in the byte number column not increasing by one in each subsequent table row, thus indicating the presence of additional bytes.

4 Security features model common to all device types

4.1 Security features for SCSI devices

4.1.1 Security associations

4.1.1.1 Principles of SAs

Before an application client and device server begin applying security functions (e.g., data integrity checking, data encryption) to messages (i.e., data that is transferred in either direction between them), they perform a security protocol to create at least one SA (see 4.1.1.3). The result of the SA creation protocol is two sets of SA parameters (see 4.1.1.2), one that is maintained by the application client and one that is maintained by the device server.

In this model, SAs decouple the process of creating a security relationship from its usage in processing security functions. This decoupling allows either the creation or the usage of an SA to be upgraded in response to changing security threats without requiring both processes to be upgraded simultaneously.

Figure 2 shows the relationship between application clients and device servers with respect to SAs.

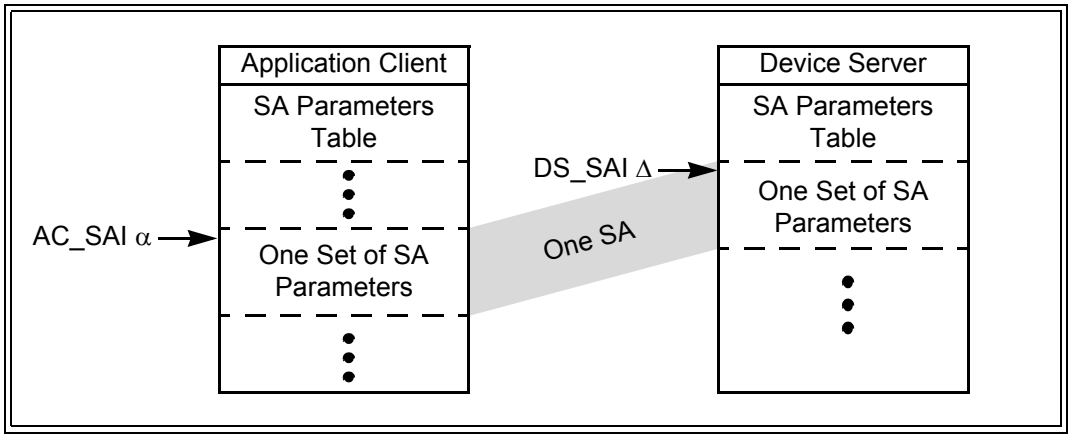

Figure 2 — SA relationships

In both the application client and the device server, the SA parameters are modeled as being stored in an indexed array and the SAI (i.e., the AC_SAI or the DS_SAI in figure 2) identifies one set of SA parameters within that array. The application client and device server are not required to store the parameters for any given SA in the same array locations. In order to support this implementation flexibility, a single SA is modeled as having two different SAI values (i.e., one for the application client and one for the device server).

The device server shall maintain a single SA parameters table for all I_T nexuses.

SAs shall not be preserved across a power cycle, hard reset, or logical unit reset. SAs shall not be affected by an I_T nexus loss.

4.1.1.2 SA parameters

Each SAI shall identify one set of SA parameters that includes at least the SA parameters defined in table 3. Individual security protocols define how the SA parameters are generated and/or used by that security protocol.

Table 3 — Minimum SA parameters (part 1 of 3)

Name	Description	Size (bytes) [a]		Scope [b]
		Min.	Max.	
SA parameters that identify and manage the SA.				
AC_SAI	The SAI used by the application client to identify the SA. [c]	4	4	Public
DS_SAI	The SAI used by the device server to identify the SA. [c]	4	4	Public
TIMEOUT	The number of seconds that may elapse after the completion of an SA access operation (i.e., SA creation or SA usage by a command) before the device server should discard the state associated with this SA (e.g., the SA parameters). If SA state is discarded because no SA access operations are received during the specified interval, then the device server shall respond to further attempts to access the SA as if the SA had never been created. This parameter shall not be set to zero.	4	4	Public
SA parameters that are incorporated in messages to prevent message replay attacks.				
AC_SQN	A sequence number that is incremented for each response message received by an application client on which a security function is performed and used to detect replay attacks (see A.4).	8	8	Public
DS_SQN	A sequence number that is incremented for each request message received by a device server on which a security function is performed and used to detect replay attacks.	8	8	Public

[a] These size values are guidelines. Specific security protocols may place more exacting size requirements on SA parameters.

[b] Public SA parameters may be transferred outside a SCSI device unencrypted. Secret SA parameters shall not be transferred outside a SCSI device. Fields within a protocol specific SA parameter are Shared or Secret as defined by the applicable SA creation protocol.

[c] SAI values between 0 and 255, inclusive, are reserved.

[d] Nonce SA parameters shall be at least half the size of the KEY_SEED SA parameter.

[e] The number of bits of entropy in the KEY_SEED SA parameter should be as close to the number of bits in the KEY_SEED SA parameter as possible (see RFC 3766).

Table 3 — Minimum SA parameters (part 2 of 3)

Name	Description	Size (bytes) [a]		Scope [b]
		Min.	Max.	
SA parameters that are used by security functions to derive the secret keys that are applied to messages (e.g., for encryption).				
AC_NONCE [d]	A random nonce value that is generated by the application client and used as an input to the key derivation security algorithm specified by the KDF_ID SA parameter during the derivation of an encryption key.	16	64	Public
DS_NONCE [d]	A random nonce value that is generated by the device server and used as an input to the key derivation security algorithm specified by the KDF_ID SA parameter during the derivation of an encryption key.	16	64	Public
KEY_SEED [e]	A value that is known only to the application client and device server that are creating this SA that in combination with the applicable nonce is used to derive the KEYMAT SA parameter. The KEY_SEED SA parameter shall be set to zero as part of completing the SA creation function.	16	64	Secret
KDF_ID	A security algorithm (see 4.1.6) coded value that identifies the KDF used by the application client and device server.	4	4	Public
SA parameters that are used by security functions to secure messages between the application client and device server.				
KEYMAT	A value that is known only to the application client and device server that are participating in this SA that may be subdivided into one or more key values that are used in security functions that secure messages. The contents of KEYMAT depend on the USAGE_TYPE SA parameter value.	14	1 024	Secret

[a] These size values are guidelines. Specific security protocols may place more exacting size requirements on SA parameters.

[b] Public SA parameters may be transferred outside a SCSI device unencrypted. Secret SA parameters shall not be transferred outside a SCSI device. Fields within a protocol specific SA parameter are Shared or Secret as defined by the applicable SA creation protocol.

[c] SAI values between 0 and 255, inclusive, are reserved.

[d] Nonce SA parameters shall be at least half the size of the KEY_SEED SA parameter.

[e] The number of bits of entropy in the KEY_SEED SA parameter should be as close to the number of bits in the KEY_SEED SA parameter as possible (see RFC 3766).

Table 3 — Minimum SA parameters (part 3 of 3)

Name	Description	Size (bytes) [a]		Scope [b]
		Min.	Max.	
SA parameters that are used by SA management functions.				
USAGE_TYPE	A coded value (see table 4) that indicates how the SA is used.	2	2	Public
USAGE_DATA	Information associated with how the SA is used (e.g., cryptographic algorithms and key sizes). The contents of USAGE_DATA depend on the USAGE_TYPE SA parameter value.	0	1 024	Public
MGMT_DATA	SA data that is used in ways defined by the SA creation protocol to perform SA management functions (e.g., deletion of the SA).	0	1 024	Protocol specific

[a] These size values are guidelines. Specific security protocols may place more exacting size requirements on SA parameters.

[b] Public SA parameters may be transferred outside a SCSI device unencrypted. Secret SA parameters shall not be transferred outside a SCSI device. Fields within a protocol specific SA parameter are Shared or Secret as defined by the applicable SA creation protocol.

[c] SAI values between 0 and 255, inclusive, are reserved.

[d] Nonce SA parameters shall be at least half the size of the KEY_SEED SA parameter.

[e] The number of bits of entropy in the KEY_SEED SA parameter should be as close to the number of bits in the KEY_SEED SA parameter as possible (see RFC 3766).

The USAGE_TYPE SA parameter (see table 4) provides an indication of how the SA is to be used.

Table 4 — USAGE_TYPE SA parameter

Code [a]	Description	USAGE_TYPE SA parameter		Reference
		Usage model	Description	
0000h to 0080h	Reserved			
0081h	Tape data encryption	ESP-SCSI [b]	None [c]	SSC-4
0082h to 8000h	Reserved			
8001h	Obsolete			
8002h to FFFFh	Reserved			

[a] USAGE_TYPE values between 8000h and CFFFh inclusive place additional constraints on how an SA is to be created as described in 5.3.5.14.

[b] ESP-SCSI usage is defined in 4.1.5.

[c] The USAGE DATA LENGTH field in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 5.3.5.14) shall contain zero.

4.1.1.3 Creating an SA

The SECURITY PROTOCOL IN command (see SPC-5) and SECURITY PROTOCOL OUT command (see SPC-5) security protocols shown in table 5 are used to create SAs. The process of creating an SA establishes the SA parameter (see 4.1.1.2) values as follows:

a) initial values for:
   A) AC_SQN set as described in 4.1.3.9; and
   B) DS_SQN set as described in 4.1.3.9;
b) unchanging values for the lifetime of the SA:
   A) AC_SAI;
   B) DS_SAI;
   C) TIMEOUT;
   D) KDF_ID;
   E) KEYMAT;
   F) USAGE_TYPE;
   G) USAGE_DATA; and
   H) MGMT_DATA;
   and
c) values that are zero (see 4.1.3.9) upon completion of SA creation:
   A) KEY_SEED;
   B) AC_NONCE; and
   C) DS_NONCE.

Table 5 — Security protocols that create SAs

Security Protocol Code	Description	References
40h	SA creation capabilities	5.2
41h	IKEv2-SCSI	4.1.3 and 5.3

4.1.2 Key derivation functions

4.1.2.1 KDFs overview

A KDF produces KEYMAT from in input KEY_SEED and STRING as follows:

KEYMAT = KDF(KEY_SEED, STRING)

Where:
KEY_SEED is a SA Parameter (see 4.1.1.2); and
STRING is a specified sequence of bytes.

Table 6 summarizes the KDFs defined by this standard.

Table 6 — KDFs summary

Security Algorithm Code (see table 25)	Description	Reference
8002 0002h	IKEv2-based iterative HMAC KDF based on SHA-1	4.1.2.3
8002 0005h	IKEv2-based iterative HMAC KDF based on SHA-256	4.1.2.3
8002 0006h	IKEv2-based iterative HMAC KDF based on SHA-384	4.1.2.3
8002 0007h	IKEv2-based iterative HMAC KDF based on SHA-512	4.1.2.3
8002 0004h	IKEv2-based iterative KDF based on AES-128 in XCBC mode	4.1.2.4

4.1.2.2 IKEv2-based iterative KDF

To produce a sufficient number of bits in KEYMAT, the IKEv2-based (see RFC 7296) iterative KDF uses the following equation:

$KEYMAT = T_1 || T_2 || T_3 || T_4 || \ldots || T_N$

Where:

T_1 = KDF(KEY_SEED, STRING || 01h);
T_2 = KDF(KEY_SEED, T_1 || STRING || 02h);
T_3 = KDF(KEY_SEED, T_2 || STRING || 03h);
T_4 = KDF(KEY_SEED, T_3 || STRING || 04h);
… = …
T_N = KDF(KEY_SEED, T_{N-1} || STRING || Nh);
KDF is the function defined in 4.1.2.1;
KEY_SEED is a SA Parameter (see 4.1.1.2); and
STRING is a specified sequence of bytes.

Protocols that use the IKEv2-based iterative KDF to generate KEYMAT should ensure that the number of KEYMAT bits requested does not cause N to exceed 255. If N reaches 256, then:

1) the requested number of KEYMAT bits is not returned; and
2) the request to produce KEYMAT shall be terminated with an error.

4.1.2.3 HMAC-based KDFs

If the KDF_ID is one of those shown in table 7, the KDF is a combination of the:

a) HMAC function defined in FIPS 198-1 (see clause 2);
b) secure hash function shown in table 7 for the specified KDF_ID value; and
c) IKEv2-based iterative KDF technique (see 4.1.2.2).

The technique requires the following inputs from or related to the SA parameters:

a) AC_SAI;
b) DS_SAI;

c) AC_NONCE;
d) DS_NONCE;
e) KEY_SEED; and
f) the number of KEYMAT bits that are to be produced.

The USAGE_TYPE SA parameter and USAGE_DATA SA parameter (see 4.1.1.2) specify the KEYMAT size as part of the security protocol that performs SA creation (see 4.1.1.3).

The IKEv2-based iterative KDF technique (see 4.1.2.2) is applied with the following inputs:

a) IFUNC (see 4.1.2.2) is the HMAC function defined in FIPS 198-1 with the translation of inputs names shown in table 7;
b) KEY_SEED is the KEY_SEED SA parameter; and
c) STRING contains the concatenated contents of the following SA parameters:
   1) AC_NONCE;
   2) DS_NONCE;
   3) AC_SAI; and
   4) DS_SAI.

Table 7 — HMAC-based KDFs

FIPS 198-1 inputs selected by KDF_ID	KDF_ID (see table 6)			
	8002 0002h	**8002 0005h**	**8002 0006h**	**8002 0007h**
H (i.e., hash function)	SHA-1 (see table 8)	SHA-256 (see table 8)	SHA-384 (see table 8)	SHA-512 (see table 8)
B (i.e., hash input block size) [a]	64	64	128	128
L (i.e., hash output block size) [a, b]	20	32	48	64
K (i.e., key)	KEY_SEED SA parameter			
text	STRING as defined in this subclause and used in 4.1.2.2			

[a] In accordance with FIPS 198-1, all sizes are shown in bytes.
[b] The HMAC-based KDFs defined by this standard do not truncate (i.e., the truncation described in FIPS 198-1 does not apply).

Details of the hash functions that act as inputs to the FIPS 198-1 HMAC function are shown in table 8.

Table 8 — Hash functions used by HMAC based on KDF_ID

KDF_ID (see table 6)	Function	Description
8002 0002h	SHA-1	HMAC input *H* is the SHA-1 secure hash function defined in FIPS 180-4 (see clause 2).
8002 0005h	SHA-256	HMAC input *H* is the SHA-256 secure hash function defined in FIPS 180-4.
8002 0006h	SHA-384	HMAC input *H* is the SHA-384 secure hash function defined in FIPS 180-4.
8002 0007h	SHA-512	HMAC input *H* is the SHA-512 secure hash function defined in FIPS 180-4.

4.1.2.4 AES-XCBC-PRF-128 IKEv2-based iterative KDF

If the KDF_ID is 8002 0004h, the KDF is a combination of:

a) the AES-XCBC-PRF-128 secure hash function defined in RFC 4434 (see clause 2) and RFC 3566; and
b) the IKEv2-based iterative KDF technique (see 4.1.2.2).

The technique requires the following inputs from or related to the SA parameters:

a) AC_SAI;
b) DS_SAI;
c) AC_NONCE;
d) DS_NONCE;
e) KEY_SEED; and
f) the number of KEYMAT bits that are to be produced.

The IKEv2-based iterative KDF (see 4.1.2.2) is applied with the following inputs:

a) IFUNC (see 4.1.2.2) is the AES-XCBC-PRF-128 secure hash function with the translation of inputs names shown in table 9;
b) KEY_SEED is the KEY_SEED SA parameter; and
c) STRING contains the concatenated contents of the following SA parameters:
   1) AC_NONCE;
   2) DS_NONCE;
   3) AC_SAI; and
   4) DS_SAI.

Table 9 — RFC 3566 parameter translations for the KDF based on AES-XCBC-PRF-128

RFC 3566 Parameter	Translation
K (i.e., key)	KEY_SEED SA parameter
M (i.e., message)	STRING as defined in this subclause and used in 4.1.2.2

4.1.3 Using IKEv2-SCSI to create an SA

4.1.3.1 Overview

The IKEv2-SCSI protocol is a subset of the IKEv2 protocol (see RFC 7296) that this standard defines for use in the creation and maintenance of an SA.

An IKEv2-SCSI SA creation transaction shall only be initiated by the application client.

The IKEv2-SCSI protocol creates the following pair of IKE SAs (see RFC 7296):

a) an SA that protects data transferred from the application client to the device server; and
b) an SA that protects data transferred from the device server to the application client.

An IKEv2-SCSI SA creation transaction consists of the following steps:

1) **Device Server Capabilities step** (see 4.1.3.5): the application client determines the device server's cryptographic capabilities;
2) **Key Exchange step** (see 4.1.3.6): the application client and device server:
   A) perform a key exchange;
   B) determine SAIs;
   C) generate the shared keys used for SA management (e.g., SA creation and deletion) (see 4.1.3.8); and
   D) may complete the generation of the SA (see 4.1.3.9);
   and
3) **Authentication step** (see 4.1.3.7): unless omitted by application client and device server negotiations in the previous steps, the application client and device server:
   A) authenticate:
      a) each other;
      b) the key exchange; and
      c) the capability selection;
      and
   B) complete the generation of the SA (see 4.1.3.9).

The values in the SECURITY PROTOCOL field and the SECURITY PROTOCOL SPECIFIC field in the SECURITY PROTOCOL IN command (see SPC-5) and SECURITY PROTOCOL OUT command (see SPC-5) identify the step of the IKEv2-SCSI protocol (see 5.3.2).

The Key Exchange step and the Authentication step depend on the results from the Device Server Capabilities step in order to create an SA. During the Key Exchange step, the application client and device server perform independent computations to construct the following sets of shared keys:

a) shared keys that are used by the Authentication step;
b) shared keys that are used by the Authentication step and to delete the SA; and
c) shared keys that are used by SCSI usage type specific operations that obtain security from the generated SA.

More details about these shared keys are provided in 4.1.3.4.

An application client may or may not:

a) proceed to the Key Exchange step after the Device Server Capabilities step; or
b) perform a separate Device Server Capabilities step for each IKEv2-SCSI SA creation transaction.

If the device server's capabilities have changed since the Device Server Capabilities step, the Authentication step returns an error and the Key Exchange step may return an error.

Changes in the device server's capabilities do not take effect until at least one application client has been notified of the new capabilities via the parameter data returned by the Device Server Capabilities step.

After a Device Server Capabilities step, the application client performs SA creation by sending a sequence of two or four IKEv2-SCSI commands over a single I_T_L nexus to the device server. The following commands constitute an IKEv2-SCSI CCS:

a) if the Authentication step is skipped (see 4.1.3.3.4):
   1) a Key Exchange step SECURITY PROTOCOL OUT command (see 4.1.3.6.2); and
   2) a Key Exchange step SECURITY PROTOCOL IN command (see 4.1.3.6.3);

   or
b) if the Authentication step is performed:
   1) a Key Exchange step SECURITY PROTOCOL OUT command (see 4.1.3.6.2);
   2) a Key Exchange step SECURITY PROTOCOL IN command (see 4.1.3.6.3);
   3) an Authentication step SECURITY PROTOCOL OUT command (see 4.1.3.7.2); and
   4) an Authentication step SECURITY PROTOCOL IN command (see 4.1.3.7.3).

The device server shall process each command in the IKEv2-SCSI CCS to completion before returning status. While a command in the IKEv2-SCSI CCS is being processed by the device server, the application client may use the REQUEST SENSE command (see 4.1.4) to ascertain the device server's progress for the command.

If an error is encountered, the device server or application client may abandon the IKEv2-SCSI CCS before the SA is created (see 4.1.3.10).

The device server shall maintain state for the IKEv2-SCSI CCS on a given I_T_L nexus from the time the Key Exchange step SECURITY PROTOCOL OUT command is completed with GOOD status until one of the following occurs:

a) the IKEv2-SCSI CCS completes successfully;
b) the IKEv2-SCSI CCS is abandoned as described in 4.1.3.10;
c) the SA being created by the IKEv2-SCSI CCS is deleted as described in 4.1.3.11;
d) the number of seconds specified in the IKEV2-SCSI PROTOCOL TIMEOUT field of the IKEv2-SCSI Timeout Values payload (see 5.3.5.15) in the Key Exchange step SECURITY PROTOCOL OUT parameter data elapses and none of the following commands have been received:
   A) the next command in the IKEv2-SCSI CCS; or
   B) a REQUEST SENSE command;

   or
e) one of the following event related SCSI device conditions (see SAM-5) occurs:
   A) power cycle;
   B) hard reset;
   C) logical unit reset; or
   D) I_T nexus loss.

If the device server receives a SECURITY PROTOCOL OUT command or SECURITY PROTOCOL IN command with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) on an I_T_L nexus other than the one for which IKEv2-SCSI CCS state is being maintained, then:

a) an additional IKEv2-SCSI CCS may be started; or
b) the device server may terminate the command with CHECK CONDITION status, with the sense key set to ABORTED COMMAND, and the additional sense code set to CONFLICTING SA CREATION REQUEST.

Except for the PERSISTENT RESERVE OUT command (see SPC-5) and the cases described in this subclause, a device server that is maintaining a IKEv2-SCSI CCS state on a particular I_T_L nexus shall not alter its processing of new commands received on that I_T_L nexus.

If all of the following conditions are true:

a) the device server includes the following algorithm descriptors in the IKEv2-SCSI SA Creation Capabilities payload (see 5.3.5.12) in the parameter data returned by the Device Server Capabilities step SECURITY PROTOCOL IN command (see 4.1.3.5):
   A) an SA_AUTH_OUT algorithm descriptor (see 5.3.6.6) with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE; and
   B) an SA_AUTH_IN algorithm descriptor (see 5.3.6.6) with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE;
   and
b) the application client sends the following algorithm descriptors to the device server in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) in the Key Exchange step SECURITY PROTOCOL OUT command (see 4.1.3.6.2):
   A) an SA_AUTH_OUT algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE; and
   B) an SA_AUTH_IN algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE,

then:

a) the Authentication step is skipped;
b) the IKEv2-SCSI CCS consists of the two Key Exchange step commands;
c) the device server requires the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 5.3.5.14) to be present in the parameter data sent by the Key Exchange step SECURITY PROTOCOL OUT command;
d) the device server returns the IKEv2-SCSI SAUT Cryptographic Algorithms payload in the parameter data returned by the Key Exchange step SECURITY PROTOCOL IN command; and
e) SA creation occurs upon the completion of the Key Exchange step.

Operation of an IKEv2-SCSI CCS depends on SA_AUTH_NONE being used in both the Authentication step SECURITY PROTOCOL OUT command and the Authentication step SECUIRTY PROTOCOL IN command, or SA_AUTH_NONE not being used by either command. Processing requirements placed on the SECURITY PROTOCOL OUT command during the Key Exchange step (see 5.3.6.6) ensure that this dependency is maintained.

If no other errors are detected and any of the following conditions are true:

a) the device server does not include an SA_AUTH_OUT algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE in the IKEv2-SCSI SA Creation Capabilities payload in the parameter data returned by the Device Server Capabilities step SECURITY PROTOCOL IN command;
b) the device server does not include an SA_AUTH_IN algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE in the IKEv2-SCSI SA Creation Capabilities payload in the parameter data returned by the Device Server Capabilities step SECURITY PROTOCOL IN command; or
c) the application client does not set the ALGORITHM IDENTIFIER field to SA_AUTH_NONE in the SA_AUTH_OUT algorithm descriptor and the SA_AUTH_IN algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload sent to the device server in the Key Exchange step SECURITY PROTOCOL OUT command,

then:

a) the Authentication step is processed;
b) the IKEv2-SCSI CCS consists of the two Key Exchange step commands and two Authentication step commands;
c) the device server requires the IKEv2-SCSI SAUT Cryptographic Algorithms payload to be absent from the parameter data sent by the Key Exchange step SECURITY PROTOCOL OUT command;
d) the device server omits the IKEv2-SCSI SAUT Cryptographic Algorithms payload from the parameter data returned by the Key Exchange step SECURITY PROTOCOL IN command;
e) the device server requires the IKEv2-SCSI SAUT Cryptographic Algorithms payload to be present in the parameter data sent by the Authentication step SECURITY PROTOCOL OUT command;
f) the device server returns the IKEv2-SCSI SAUT Cryptographic Algorithms payload in the parameter data returned by the Authentication step SECURITY PROTOCOL IN command; and
g) SA creation occurs upon the completion of the Authentication step.

SA participants should perform the Authentication step unless man-in-the-middle attacks (see A.4) are not of concern or are prevented by a means outside the scope of this standard (e.g., physical security of the transport).

Omission of the Authentication step provides no defense against a man-in-the-middle adversary that is capable of modifying SCSI commands. Such an adversary is able to insert itself as an intermediary on the created SA without knowledge of the SA participants, thereby completely subverting the intended security. Omission of the Authentication step is only appropriate in environments where the absence of such adversaries is assured by other means.

EXAMPLE – Environments where it is appropriate to skip the Authentication step include those that protect data transfers using:

a) a direct physical connection between the systems that contain the application client and the device server; or
b) end-to-end security in the SCSI transport protocol (e.g., FC-SP-2).

4.1.3.2 IKEv2-SCSI Protocol summary

This subclause summarizes the IKE-v2-SCSI payloads (see 5.3.5) that are exchanged between an application client and a device server during all steps of an IKEv2-SCSI SA creation transaction using message diagrams. Each IKEv2-SCSI step (see 4.1.3.1) is shown in a separate figure. The contents of a payload (e.g., Key Exchange) may not be the same in both directions of transfer.

Figure 3 shows the Device Server Capabilities step (see 4.1.3.5). The Device Server Capabilities step consists of a SECURITY PROTOCOL IN command carrying an IKEv2-SCSI SA Creation Capabilities payload (see 5.3.5.12). The IKEv2-SCSI header is not used.

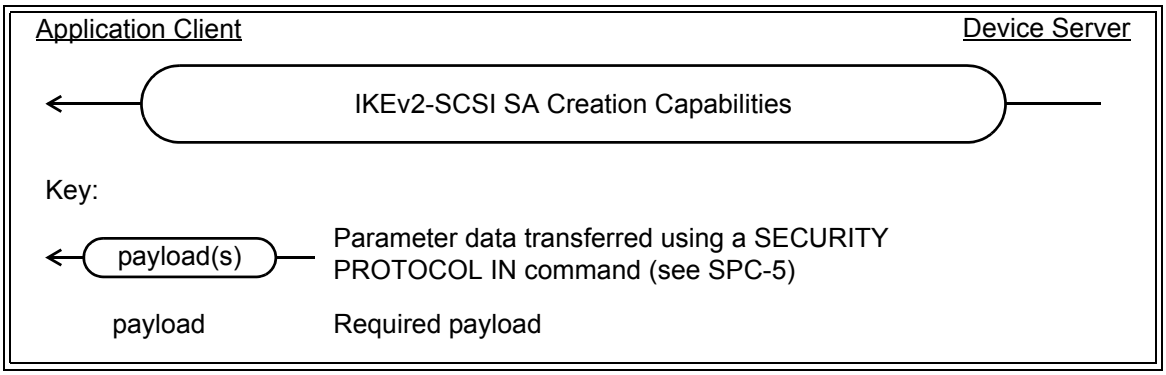

Figure 3 — IKEv2-SCSI Device Server Capabilities step

The IKEv2-SCSI SA Creation Capabilities payload indicates the device server's capabilities for SA creation.

Figure 4 shows the Key Exchange step (see 4.1.3.6). The Key Exchange step consists of a SECURITY PROTOCOL OUT command followed by a SECURITY PROTOCOL IN command.

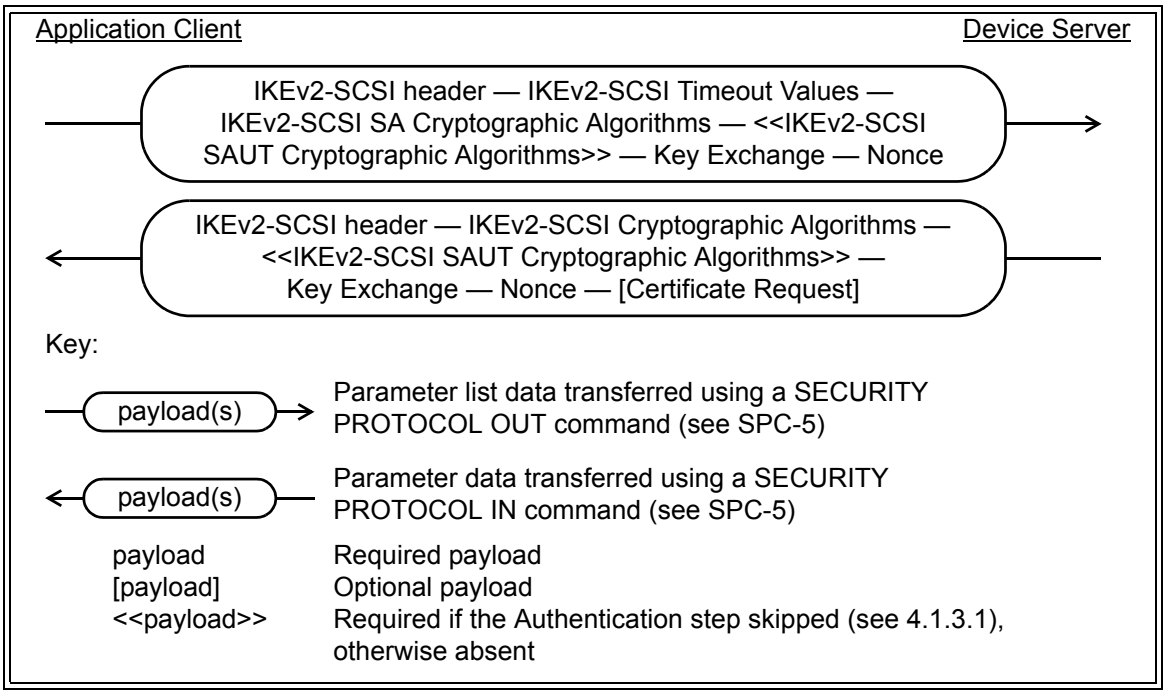

Figure 4 — IKEv2-SCSI Key Exchange step

The IKEv2-SCSI Timeout Values payload (see 5.3.5.15) contains timeouts for SA creation and usage.

The IKEv2-SCSI SA Cryptographic Algorithms payloads (see 5.3.5.13) are used to select and agree on the cryptographic algorithms used for creating the SA.

If the Authentication step is skipped (see 4.1.3.1), the IKEv2-SCSI SAUT Cryptographic Algorithms payloads (see 5.3.5.14) are used to select and agree on usage of the SA and the cryptographic algorithms used by the created SA. If the Authentication step is processed (i.e., not skipped), the SA usage and algorithms selection is performed during the Authentication step.

The Key Exchange payload (see 5.3.5.3) and Nonce payload (see 5.3.5.8) are part of the key and nonce exchanges that are used to generate the IKEv2-SCSI keys and SA keys.

The Certificate Request payload or payloads (see 5.3.5.6) enables the device server to request a certificate from the application client. If the Authentication step is being skipped (see 4.1.3.1), the device server shall not include any Certificate Request payloads in the parameter data. Use of the Certificate Request payload is described in 4.1.3.3.3.4.

Figure 5 shows the Authentication step (see 4.1.3.7). The Authentication step consists of a SECURITY PROTOCOL OUT command followed by a SECURITY PROTOCOL IN command.

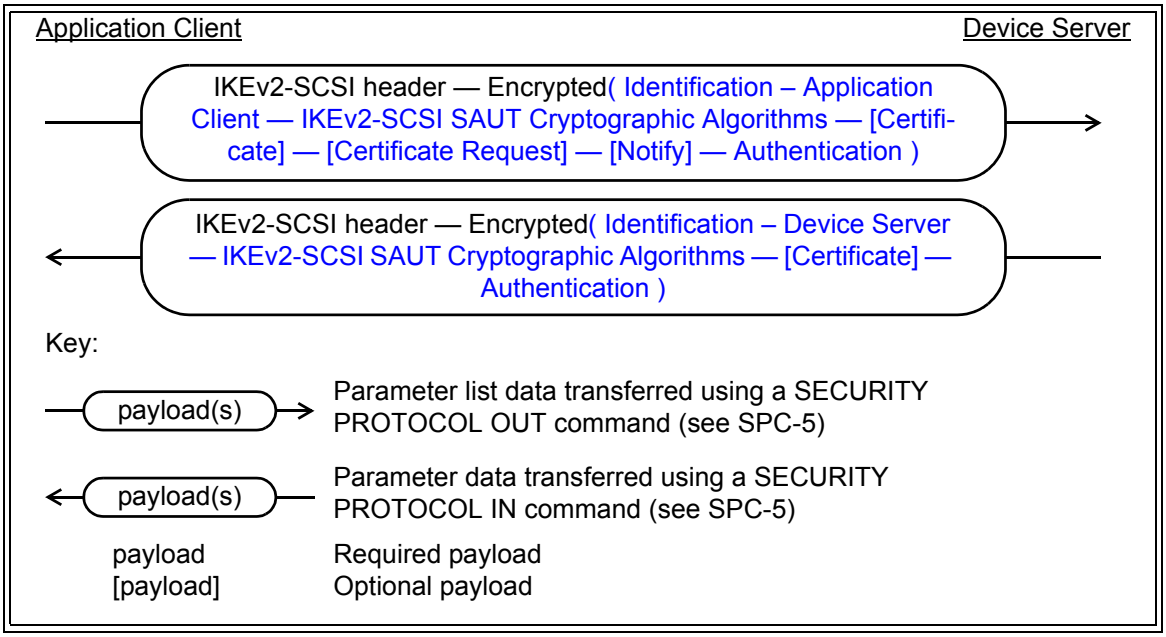

Figure 5 — IKEv2-SCSI Authentication step

An Encrypted payload (see 5.3.5.11) contains all other Authentication step payloads that are protected using the cryptographic algorithms determined by the IKEv2-SCSI SA Cryptographic Algorithms payloads (see 5.3.5.13) in the Key Exchange step (see figure 4).

The Identification payloads (see 5.3.5.4) contain the identities to be authenticated. These identities are not required to be SCSI names or identifiers.

The IKEv2-SCSI SAUT Cryptographic Algorithms payloads (see 5.3.5.14) are used to select and agree on usage of the SA and the cryptographic algorithms used by the created SA.

The Certificate payload or payloads (see 5.3.5.5) respond to the Certificate Request payload(s) sent by the device server in the Key Exchange step SECURITY PROTOCOL IN command.

The Certificate Request payload or payloads (see 5.3.5.6) allows an application client to request the delivery of a Certificate payload (see 5.3.5.5) in the parameter data for the Authentication step SECURITY PROTOCOL IN command (see 4.1.3.3.3.4).

The Notify payload (see 5.3.5.9) provides a means for the application client to inform the device server that this is the only SA being used between them, and that the device server should discard state for any other SAs created by the same application client.

The Authenticate payloads (see 5.3.5.7) authenticate not only the SA participants, but also the entire protocol sequence (e.g., the Authenticate payloads prevent a man-in-the-middle attack from succeeding).

Figure 6 shows the Delete operation (see 4.1.3.11). The Delete operation consists of a SECURITY PROTOCOL OUT command.

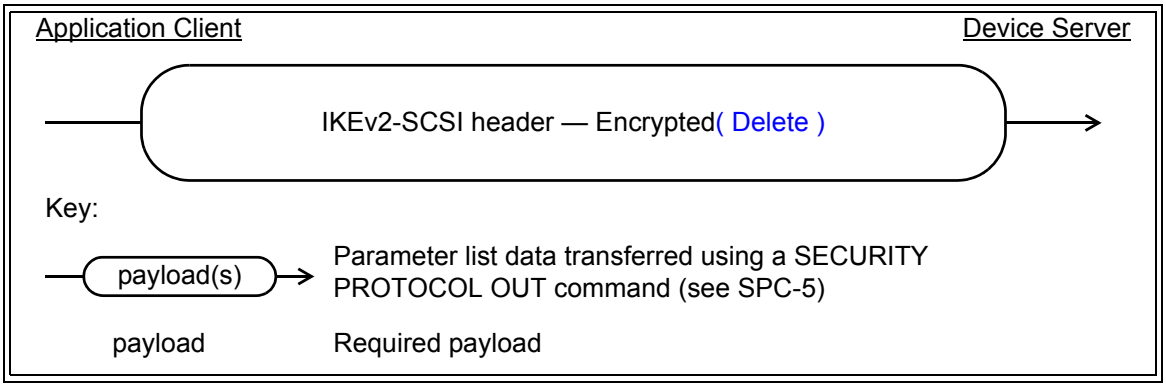

Figure 6 — IKEv2-SCSI Delete operation

An Encrypted payload (see 5.3.5.11) contains the Delete payload that is protected using the cryptographic algorithms determined by the IKEv2-SCSI SA Cryptographic Algorithms payloads (see 5.3.5.13) in the Key Exchange step that was used to create the SA.

The Delete payload (see 5.3.5.10) specifies the SA to be deleted.

4.1.3.3 IKEv2-SCSI Authentication

4.1.3.3.1 Overview

IKEv2-SCSI authentication includes these security functions:

a) the application client and device server each establish an identity by demonstrating knowledge of a secret authentication key associated with that identity;
b) the application client demonstrates knowledge of the current device server capability information; and
c) the application client and device server check the integrity of the current IKEv2-SCSI CCS.

An IKEv2 SCSI authentication algorithm accomplishes these functions by generating and verifying authentication data based on a concatenation of bytes that includes device server capability information and the specified portion of the IKEv2-SCSI parameter data (see 5.3.5.7) from the IKEv2-SCSI Key Exchange step (see 4.1.3.6).

An authentication key associated with an identity is used to generate authentication data. IKEv2-SCSI transfers the authentication data in the AUTHENTICATION DATA field of the IKEv2-SCSI Authentication payload (see 5.3.5.7). The recipient of an IKEv2-SCSI Authentication payload uses a verification key associated with the identity to verify the authentication data. The identity is:

a) transferred in the IDENTIFICATION DATA field of the appropriate IKEv2-SCSI Identification payload (see 5.3.5.4); or
b) obtained from a certificate transferred in the IKEv2-SCSI Certificate payload (see 5.3.5.5).

IKEv2-SCSI Authentication is bidirectional (i.e., both the application client and the device server authenticate). IKEv2-SCSI Authentication is skipped when the application client and device server agree to do so during the Key Exchange step (see 4.1.3.3.4).

The following IKEv2-SCSI Authentication methods are defined:

a) **pre-shared key** (see 4.1.3.3.2): the authentication key is also used as the verification key; and
b) **digital signature** (see 4.1.3.3.3): the verification key and a different authentication key form a public/private key pair. The authentication data is a digital signature.

Certificates and the IKEv2-SCSI Certificate payload may be used to provide verification keys for digital signatures to application clients and device servers.

4.1.3.3.2 Pre-shared key authentication

Pre-shared key authentication uses a single cryptographic algorithm to both generate and verify authentication data. A pre-shared key is associated with an identity that is transferred in the IDENTIFICATION DATA field of the appropriate Identification payload (see 5.3.5.4). The pre-shared key serves as both the authentication key and the verification key for the identity.

NOTE 1 - A pre-shared key that is not kept secret may compromise the security properties of IKEv2-SCSI.

If pre-shared key authentication is used, then the pre-shared key is the key for the cryptographic algorithm. Authentication data is generated by applying the cryptographic algorithm with this key to the input data (e.g., the applicable concatenation of bytes described in 5.3.5.7).

Verification of the authentication data shall consist of:

1) computing the expected contents of the AUTHENTICATION DATA field of the Authentication payload (see 5.3.5.7) using the input data and a verification key associated with the identity received in the Identification payload (see 5.3.5.4); and
2) comparing the expected contents to the actual contents of the AUTHENTICATION DATA field.

Verification is successful if the expected contents match the actual contents, otherwise verification is not successful.

The pre-shared key requirements in RFC 7296 shall apply to IKEv2-SCSI pre-shared keys, including the following requirements on interfaces for provisioning pre-shared keys:

a) ASCII strings of at least 64 bytes shall be supported;
b) a null terminator shall not be added to any input before it is used as a pre-shared key;
c) a hexadecimal ASCII encoding of the pre-shared key shall be supported; and
d) ASCII encodings other than hexadecimal may be supported. Support for any such encoding shall include specification of the algorithm for translating the encoding to a binary string as part of the interface.

The following requirements for pre-shared keys apply in addition to those found in RFC 7296:

a) a pre-shared key shall be associated with one identity;
b) the same pre-shared key shall not be used to authenticate both an application client and a device server;
c) the same pre-shared key should not be used for a group of application clients or a group of device servers;
d) information about the size of the pre-shared key shall be stored at the same time that the pre-shared key is stored; and
e) the means for provisioning pre-shared keys are outside the scope of this standard.

4.1.3.3.3 Digital signature authentication

4.1.3.3.3.1 Overview

Digital signature authentication uses a matched pair of signature and verification cryptographic algorithms to generate and verify authentication data that is a digital signature. A public/private key pair is associated with an identity. The private key is used as the authentication key for the identity. The public key is used as the verification key for the identity.

NOTE 2 - A private authentication key that is not kept secret may compromise the security properties of IKEv2-SCSI.

If digital signature authentication is used, then the private key is the key for the signature algorithm. A digital signature is generated by applying the signature algorithm with this private key to the input data (e.g., the applicable concatenation of bytes described in 5.3.5.7).

Verification of the digital signature shall consist of using the public verification key associated with the identity and the input data to verify the digital signature received as the contents of the AUTHENTICATION DATA field of the Authentication payload (see 5.3.5.7). Verification is successful if the digital signature is a valid digital signature over the input data, otherwise verification is not successful.

The method used by an application client or a device server to obtain a private authentication key are outside the scope of this standard. An identity and associated public verification key are obtained as follows:

a) if certificates are used for digital signature authentication, then the identity and the associated public verification key are obtained from a certificate transferred in the first IKEv2-SCSI Certificate payload (see 4.1.3.3.3.4); or
b) if certificates are not used for digital signature authentication, then the identity is transferred in the IDENTIFICATION DATA field of the appropriate IKEv2-SCSI Identification payload (see 5.3.5.4) and the public verification key may be:
   A) transferred as a raw RSA key in an IKEv2-SCSI Certificate payload (see 5.3.5.5); or
   B) obtained by means that are outside the scope of this standard.

If certificates are not used for digital signature authentication, the association between the identity and the public key should be verified by means outside the scope of this standard.

4.1.3.3.3.2 Certificates and digital signature authentication

A certificate (see RFC 5280 and RFC 6818) is a data structure that contains:

a) an identity;
b) a public key for that identity;
c) additional relevant information that may constrain use of the public key;
d) the identity of a certification authority (see RFC 5280 and RFC 6818); and
e) a digital signature generated by that certification authority.

If the identity and associated public key used to verify a digital signature are obtained from a certificate, then the certification path from the certificate to a trust anchor should be validated (see RFC 5280 and RFC 6818). If certification path validation is not successful, verification of the digital signature for that identity shall fail independent of whether the digital signature is valid.

The method used by an application client or a device server to obtain a trust anchor are outside the scope of this standard.

4.1.3.3.3.3 Example of certificate use for digital signature authentication

An example of certificate use involves an application client or device server that trusts a certification authority. Based on this trust, the public key of that certification authority is used to validate a certificate presented as part of authentication. Successful validation of that certificate establishes that the public key in that certificate is associated with the identity in the certificate. That public key is then used to verify the digital signature in the Authentication payload (see 5.3.5.7).

In this example, providing a certificate as part of the IKEv2-SCSI Authentication step (see 4.1.3.7) allows a single certification authority public key to serve as a trust anchor (see RFC 5280 and RFC 6818) for verification of digital signatures for any identity that has been issued a certificate by that certification authority, so that a public key for each identity does not have to be obtained by other means.

Validating a certificate includes multiple checks beyond verifying the signature, and the validation may traverse a certification path composed of multiple certificates (see RFC 5280 and RFC 6818).

4.1.3.3.3.4 Handling of the Certificate Request payload and the Certificate payload

As detailed in this subclause, a Certificate Request payload (see 5.3.5.6) in one set of parameter data requests the delivery of a Certificate payload (see 5.3.5.5) in the next set of parameter data transferred. The purpose of these IKEv2-SCSI protocol elements is as follows:

a) each SA participant is allowed to require the delivery of a Certificate payload by the other SA participant for use in authentication; and
b) each Certificate Request payload indicates the trust anchors list (see RFC 7296) used by the device server or application client when PKI-based Authentication is being used with certificates that are not self signed (see RFC 5280 and RFC 6818).

The presence of one or more Certificate Request payloads in the Key Exchange step SECURITY PROTOCOL IN command (see 4.1.3.6.3) parameter data indicates that the device server requires the application client to include a Certificate payload in the parameter list for the Authentication step SECURITY PROTOCOL OUT command (see 4.1.3.7.2).

The presence of one or more Certificate Request payloads in the Authentication step SECURITY PROTOCOL OUT command parameter list specifies that the application client requires the device server to return a Certificate payload in the parameter data for the Authentication step SECURITY PROTOCOL IN command (see 4.1.3.7.3).

If any Certificate payloads are included in the parameter data, the first Certificate payload shall contain the public key used to verify the Authentication payload. Additional Certificate payloads may be used to assist in establishing a certification path from the certificate in the first payload to a trust anchor (see RFC 7296, RFC 5280 and RFC 6818).

The application client and device server may use different authentication methods that require or do not require the use of Certificate payloads. The presence or absence of Certificate Request payloads and Certificate payloads may vary in any of the commands described in this subclause.

4.1.3.3.4 Constraints on skipping the Authentication step

In the Device Server Capabilities step (see 4.1.3.5), the parameter data returned by the SECURITY PROTOCOL IN command (see 5.2.3.2) contains the IKEv2-SCSI SA Creation Algorithms payload (see 5.3.5.12) that contains one or more SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptors (see 5.3.6.6) and one or more SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptors.

The device server shall allow the Authentication step to be omitted (see 4.1.3.1) if:

a) the ALGORITHM IDENTIFIER field is set to SA_AUTH_NONE (see 5.3.6.6) in one of the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptors returned in the Device Server Capabilities step; and
b) the ALGORITHM IDENTIFIER field is set to SA_AUTH_NONE in one of the SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptors returned in the Device Server Capabilities step.

The methods for configuring a device server to return SA_AUTH_NONE are outside the scope of this standard. Device servers shall not be manufactured to return SA_AUTH_NONE as an Authentication payload authentication algorithm type in the Device Server Capabilities step.

In the Key Exchange step SECURITY PROTOCOL OUT command (see 4.1.3.6.2), the application client requests that the Authentication step be omitted by setting the ALGORITHM IDENTIFIER field to SA_AUTH_NONE in:

a) the SA_AUTH_OUT cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13); and
b) the SA_AUTH_IN cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload.

To ensure adequate SA security, the application client should not select the SA_AUTH_NONE value as an Authentication payload authentication algorithm type unless:

a) an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor from the Device Server Capabilities step indicates SA_AUTH_NONE availability; and
b) the application client is configured to omit the Authentication step.

If SA_AUTH_NONE is used, IKEv2-SCSI has no protection against man-in-the-middle attacks. Enabling return of the SA_AUTH_NONE authentication algorithm type in the Device Capabilities step, and allowing an application client to select SA_AUTH_NONE in the Key Exchange step are administrative security policy decisions that absence of authentication is acceptable. Such decisions should only be made in situations where active attacks on IKEv2-SCSI are not of concern (e.g., direct attachment of a SCSI initiator device and a SCSI target device, or an end-to-end secure service delivery subsystem such as Fibre Channel secured by an end-to-end FC-SP-2 SA).

4.1.3.4 Summary of IKEv2-SCSI shared keys nomenclature and shared key sizes

The IKEv2-SCSI shared keys are named as shown in table 10.

Table 10 — IKEv2-SCSI shared key names and SA shared key names

Name	SA parameter that stores this shared key	Description
Shared keys used only during Authentication step		
SK_pi	shall not be stored in any SA parameter	Shared key used to construct the Authentication payload (see 5.3.5.7) for the SECURITY PROTOCOL OUT parameter list in the Authentication step (see 4.1.3.7.2).
SK_pr		Shared key used to construct the Authentication payload for the SECURITY PROTOCOL IN parameter data in the Authentication step.
Shared keys used: a) during IKEv2-SCSI SA creation and management; and b) for bytes in a Data-Out Buffer or Data-In Buffer		
SK_ai	MGMT_DATA	Shared key used to integrity check the Encrypted payload in the SECURITY PROTOCOL OUT parameter list in the: a) Authentication step; and b) IKEv2-SCSI Delete operation (see 4.1.3.11).
	KEYMAT	Shared key used to integrity check the contents of a Data-Out Buffer.
SK_ar	MGMT_DATA	Shared key used to integrity check the Encrypted payload (see 5.3.5.11) in the SECURITY PROTOCOL IN parameter data in the Authentication step (see 4.1.3.7.3).
	KEYMAT	Shared key used to integrity check the contents of a Data-In Buffer.
SK_ei	MGMT_DATA	Shared key used to encrypt the Encrypted payload in the SECURITY PROTOCOL OUT parameter list in the: a) Authentication step; and b) IKEv2-SCSI Delete operation.
	KEYMAT	Shared key used to encrypt the contents of a Data-Out Buffer.
SK_er	MGMT_DATA	Shared key used to encrypt the Encrypted payload in the SECURITY PROTOCOL IN parameter data in the Authentication step.
	KEYMAT	Shared key used to encrypt the contents of a Data-In Buffer.
Shared key used to construct the SA keys		
SK_d	KEY_SEED	Shared key material that is used as input to the KDF that generates the KEYMAT SA parameter bytes for the SA.

The sizes of the shared keys are determined as shown in table 11.

Table 11 — Shared key size determination

Name	Shared key size determination	
	Separate encryption and integrity checking [a, b]	**Combined mode encryption and integrity checking** [a, b]
SK_ai and SK_ar [c]	The shared key size is shown in table 68 for the value in the ALGORITHM IDENTIFIER field of the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.4) [b].	zero [d]
SK_er and SK_ei [c]	The shared key size is the value in the KEY LENGTH field of the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.2) [b].	The shared key size is the value in the KEY LENGTH field of the ENCR IKEv2-SCSI cryptographic algorithm descriptor plus the number of salt bytes shown in table 64 (see 5.3.6.2) [b].
SK_pi and SK_pr [c]	The shared key size is equal to the PRF output length (see table 66) associated with the value in the ALGORITHM IDENTIFIER field of the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13).	
SK_d		

[a] The use of combined mode encryption and integrity checking is indicated by the AUTH_COMBINED value in the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.4).

[b] For the shared keys used to create an SA, the algorithm descriptor is located in an IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13). For the shared keys used after the SA is created (i.e., the KEYMAT SA parameter), the algorithm descriptor is located in an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 5.3.5.14).

[c] To accommodate two shared keys of the specified size, the shared key length shown is doubled for the purposes of shared key generation.

[d] In combined mode encryption and integrity checking, the SK_er and SK_ei are used for both encryption and integrity checking.

4.1.3.5 Device Server Capabilities step

In the Device Server Capabilities step, the application client sends a SECURITY PROTOCOL IN command (see SPC-5) with the SECURITY PROTOCOL field set to SA creation capabilities (i.e., 40h) and the SECURITY PROTOCOL SPECIFIC field set to 0101h.

The device server returns the SECURITY PROTOCOL IN parameter data specified by the SECURITY PROTOCOL SPECIFIC field (see 5.2.2) and the parameter data (see 5.2.3.2) contains an IKEv2-SCSI SA Creation Capabilities payload (see 5.3.5.12).

In the Device Server Capabilities step, the device server shall return parameter data containing the IKEv2-SCSI cryptographic algorithm descriptors (see 5.3.6) in at least one complete row shown in table 12.

Table 12 — Device Server Capabilities step parameter data requirements

IKEv2-SCSI cryptographic algorithm descriptor					
ENCR (see 5.3.6.2)	**PRF (see 5.3.6.3)**	**INTEG (see 5.3.6.4)**	**D-H (see 5.3.6.5)**	**SA_AUTH_OUT (see 5.3.6.6)**	**SA_AUTH_IN (see 5.3.6.6)**
The ALGORITHM IDENTIFIER field set to 8001 0014h (i.e., AES-GCM) and the KEY LENGTH field set to 0010h	The ALGORITHM IDENTIFIER field set to 8002 0005h (i.e., IKEv2-use based on SHA-256)	The ALGORITHM IDENTIFIER field set to F003 0001h (i.e., AUTH_ COMBINED)	The ALGORITHM IDENTIFIER field set to 8004 000Eh (i.e., 2 048-bit MODP group (finite field D-H))	The ALGORITHM IDENTIFIER field set to 00F9 0001h (i.e., RSA Digital Signature)	The ALGORITHM IDENTIFIER field set to 00F9 0001h (i.e., RSA Digital Signature)
The ALGORITHM IDENTIFIER field set to 8001 000Ch (i.e., AES-CBC) and the KEY LENGTH field set to 0020h	The ALGORITHM IDENTIFIER field set to 8002 0007h (i.e., IKEv2-use based on SHA-512)	The ALGORITHM IDENTIFIER field set to 8003 000Eh (i.e., AUTH_HMAC_ SHA2_ 512_256)	The ALGORITHM IDENTIFIER field set to 8004 0015h (i.e., 521-bit random ECP group)	The ALGORITHM IDENTIFIER field set to 00F9 000Bh (i.e., ECDSA with SHA-512 on the P-521 curve)	The ALGORITHM IDENTIFIER field set to 00F9 000Bh (i.e., ECDSA with SHA-512 on the P-521 curve)

In the Device Server Capabilities step, the device server shall return parameter data containing one SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE and one SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE if any of the following are true:

a) the ALGORITHM IDENTIFIER field is set to SA_AUTH_NONE in one of the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptors returned in the Device Server Capabilities step; or
b) the ALGORITHM IDENTIFIER field is set to SA_AUTH_NONE in one of the SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptors returned in the Device Server Capabilities step.

The device server capabilities returned in the SECURITY PROTOCOL IN parameter data may be changed at any time by means that are outside the scope of this standard, However, such changes shall not take effect until at least one application client has been notified of the new capabilities in the parameter data returned by the Device Server Capabilities step SECURITY PROTOCOL IN command. Management applications may ensure that their device server capabilities changes take effect by sending a Device Server Capabilities step SECURITY PROTOCOL IN command to the device server after the changes have been made.

If the device server capabilities change (i.e., upon completion of the processing for a Device Server Capabilities step SECURITY PROTOCOL IN command that reported changed information in its parameter data), then the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus except the I_T nexus on which the Device Server Capabilities step SECURITY PROTOCOL IN command was received (see SAM-5), with the additional sense code set to SA CREATION CAPABILITIES DATA HAS CHANGED.

The Device Server Capabilities step participates in the negotiation to skip the Authentication step as described in 4.1.3.3.4.

NOTE 3 - The Device Server Capabilities step has no IKEv2 exchange equivalent in RFC 7296. This step replaces most of IKEv2's negotiation by having the application client obtain the supported capabilities from the device server.

4.1.3.6 IKEv2-SCSI Key Exchange step

4.1.3.6.1 Overview

The Key Exchange step consists of a Diffie-Hellman key exchange with nonces (see RFC 7296) and is accomplished as follows:

1) a SECURITY PROTOCOL OUT command (see 4.1.3.6.2);
2) a SECURITY PROTOCOL IN command (see 4.1.3.6.3); and
3) key exchange completion (see 4.1.3.6.4).

NOTE 4 - The Key Exchange step corresponds to the IKEv2 IKE_SA_INIT exchange in RFC 7296, except that determination of device server capabilities has been moved to the Device Server Capabilities step.

4.1.3.6.2 Key Exchange step SECURITY PROTOCOL OUT command

To send its key exchange message to the device server, the application client sends a SECURITY PROTOCOL OUT command (see SPC-5) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0102h. The parameter list consists of an IKEv2-SCSI header (see 5.3.4) and the following:

1) an IKEv2-SCSI Timeout Values payload (see 5.3.5.15);
2) an IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13);
3) if the Authentication step is skipped (see 4.1.3.1), an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 5.3.5.14);
4) a Key Exchange payload (see 5.3.5.3); and
5) a Nonce payload (see 5.3.5.8).

The IKEv2-SCSI Timeout Values payload contains the inactivity timeouts that apply to this IKEv2-SCSI SA creation transaction and the SA that is created.

The IKEv2-SCSI SA Cryptographic Algorithms payload selects the cryptographic algorithms from among those returned in the Device Server Capabilities step (see 4.1.3.5) to be used in the creation of the SA.

If the Authentication step is skipped, the IKEv2-SCSI SAUT Cryptographic Algorithms payload contains the following information about the SA to be created:

a) the cryptographic algorithms selected by the application client from among those returned in the Device Server Capabilities step; and
b) the usage data (see 5.3.5.14), if any, that is specific to the SA.

If the application client is unable to select a set of algorithms that are appropriate for the intended creation and usage of the SA, then the application client should not perform the Key Exchange step to request the creation of an SA.

IKEv2-SCSI SA Cryptographic Algorithms payload error checking requirements that ensure a successful negotiation of SA creation algorithms are described in 5.3.5.13 and 5.3.6.

IKEv2-SCSI SAUT Cryptographic Algorithms payload error checking requirements that ensure a successful SA creation are described in 5.3.5.14 and 5.3.6.

The Key Exchange payload contains the value sent by the application client during a Diffie-Hellman key exchange (see 5.3.5.3).

The Nonce payload contains the application client's random nonce.

4.1.3.6.3 Key Exchange step SECURITY PROTOCOL IN command

If the Key Exchange step SECURITY PROTOCOL OUT command (see 4.1.3.6.2) completes with GOOD status, then the application client sends a SECURITY PROTOCOL IN command (see SPC-5) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0102h to obtain the device server's key exchange message.

The parameter data returned by the device server in response to the SECURITY PROTOCOL IN command shall contain an IKEv2-SCSI header (see 5.3.4) and the following:

1) an IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13);
2) if the Authentication step is skipped (see 4.1.3.1), an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 5.3.5.14);
3) a Key Exchange payload (see 5.3.5.3);
4) a Nonce payload (see 5.3.5.8); and
5) zero or more Certificate Request payloads (see 5.3.5.6).

As part of processing of the Key Exchange step SECURITY PROTOCOL IN command, the device server shall:

a) associate the SECURITY PROTOCOL IN command to the last Key Exchange step SECURITY PROTOCOL OUT command received on the I_T_L nexus. If the device server is maintaining state for at least one IKEv2-SCSI CCS and the device server is unable to establish this association, then the device server shall:
   A) terminate the SECURITY PROTOCOL IN command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
   B) continue the IKEv2-SCSI CCS.

   If the device is not maintaining state for at least one IKEv2-SCSI CSS, the device server shall terminate the SECURITY PROTOCOL IN command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR;
b) return the device server’s SAI in the IKEv2-SCSI header IKE_SA DEVICE SERVER SAI field;
c) return the IKEv2-SCSI SA Cryptographic Algorithms payload containing:
   A) the SA creation cryptographic algorithms supplied by the application client in the Key Exchange step SECURITY PROTOCOL OUT command parameter list; and
   B) the device server's SAI in the SAI field (see 5.3.5.13);
d) if the Authentication step is skipped, return the IKEv2-SCSI SAUT Cryptographic Algorithms payload containing:
   A) the SA usage cryptographic algorithms supplied by the application client in the Key Exchange step SECURITY PROTOCOL OUT command parameter list; and
   B) the device server's SAI in the SAI field (see 5.3.5.14);
e) return information about the completed Diffie-Hellman exchange with the Key Exchange payload; and
f) return the device server’s random nonce in the Nonce payload.

If the Key Exchange step SECURITY PROTOCOL IN command (see 4.1.3.6.3) completes with GOOD status, then the application client should copy the device server’s SAI from the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header to the state the application client is maintaining for the IKEv2-SCSI CCS.

Except for the SAI field, the application client should compare the fields in the IKEv2-SCSI SA Cryptographic Algorithms payload and the IKEv2-SCSI SAUT Cryptographic Algorithms payload, if any, to the values sent in the Key Exchange step SECURITY PROTOCOL OUT command (see 4.1.3.6.2). If the application client detects differences in the contents of the payloads other than in the SAI field, then the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 4.1.3.10.

4.1.3.6.4 Key Exchange step completion

Before completing the Key Exchange step SECURITY PROTOCOL IN command (see 4.1.3.6.3) with GOOD status the device server shall complete the Key Exchange step as described in this subclause.

Upon receipt of GOOD status for the Key Exchange step SECURITY PROTOCOL IN command the application client should complete the Key Exchange step as described in this subclause.

If the Key Exchange step does not end with the IKEv2-SCSI CCS being abandoned (see 4.1.3.10), then the contents of the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.6) in the IKEv2-SCSI SA Cryptographic Algorithms payload specify how the shared key exchanged by the Key Exchange step SECURITY PROTOCOL OUT command and the Key Exchange step SECURITY PROTOCOL IN command is used to generate additional shared keys as follows:

a) if the ALGORITHM IDENTIFIER field in both descriptors contain SA_AUTH_NONE, then the SA participants generate the SA, including the generation of the shared keys used for SA management (e.g., SA creation and management) and the shared keys used by the created SA as defined in 4.1.3.9; or
b) if the ALGORITHM IDENTIFIER field in both descriptors contain a value other than SA_AUTH_NONE, then the SA participants generate shared keys (see 4.1.3.8.3) for the following:
   A) seeding the Authentication step generation of the shared keys used by the created SA; and
   B) SA creation and management.

4.1.3.6.5 After the Key Exchange step

Processing of the IKEv2-SCSI CCS subsequent to completion of the Key Exchange step (see 4.1.3.6.4) depends on the contents of the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.6) in the IKEv2-SCSI SA Cryptographic Algorithms payload as follows:

a) if the ALGORITHM IDENTIFIER field in both descriptors contain SA_AUTH_NONE, then processing of the IKEv2-SCSI CCS is finished and the generated SA is ready for use; or
b) if the ALGORITHM IDENTIFIER field in both descriptors contain a value other than SA_AUTH_NONE, then processing of the IKEv2-SCSI CCS continues as follows:
   A) the Authentication step is performed (see 4.1.3.7); and
   B) the SA participants generate the SA, including the generation of the shared keys used for SA management and the shared keys used by the created SA as defined in 4.1.3.9.

4.1.3.7 IKEv2-SCSI Authentication step

4.1.3.7.1 Overview

The Authentication step performs the following functions:

a) authenticates both the application client and the device server;
b) protects the previous steps of the protocol; and

c) cryptographically binds the authentication and the previous steps to the created SA.

The Authentication step is accomplished as follows:

1) a SECURITY PROTOCOL OUT command (see 4.1.3.7.2); and
2) a SECURITY PROTOCOL IN command (see 4.1.3.7.3).

The parameter data for both commands shall be encrypted and integrity protected using the algorithms and keys determined in the Key Exchange step (see 4.1.3.6).

NOTE 5 - The Authentication step corresponds to the IKEv2 IKE_AUTH exchange in RFC 7296.

4.1.3.7.2 Authentication step SECURITY PROTOCOL OUT command

To send its authentication message to the device server, the application client sends a SECURITY PROTOCOL OUT command (see SPC-5) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0103h. The parameter data consists of the IKEv2-SCSI header (see 5.3.4) and an Encrypted payload (see 5.3.5.11) that:

a) is integrity checked and encrypted in one of the following ways:
   A) using separate algorithms as follows:
      a) integrity checked using:
         A) the algorithm specified by the INTEG IKEv2-SCSI algorithm descriptor (see 5.3.6.4) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) from the Key Exchange step (see 4.1.3.6); and
         B) the SK_ai shared key (see 4.1.3.4);

         and
      b) encrypted using:
         A) the algorithm specified by the ENCR IKEv2-SCSI algorithm descriptor (see 5.3.6.2) in the IKEv2-SCSI SA Cryptographic Algorithms payload from the Key Exchange step; and
         B) the SK_ei shared key (see 4.1.3.4);

      or
   B) using a combined integrity check and encryption algorithm that uses the following (i.e., if the INTEG IKEv2-SCSI algorithm descriptor indicates AUTH_COMBINED):
      a) the algorithm specified by the ENCR IKEv2-SCSI algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload; and
      b) the SK_ei shared key with additional salt bytes as described in 4.1.3.8.1 and table 11 (see 4.1.3.4);

   and
b) contains the following:
   1) an Identification – Application Client payload (see 5.3.5.4);
   2) an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 5.3.5.14);
   3) zero or more Certificate payloads (see 5.3.5.5);
   4) zero or more Certificate Request payloads (see 5.3.5.6);
   5) zero or one Notify payload (see 5.3.5.9); and
   6) an Authentication payload (see 5.3.5.7).

Before performing any checks on data contained in the Encrypted payload, the device server shall validate the SECURITY PROTOCOL OUT command parameter data as follows:

a) the device server shall compare the IKE_SA APPLICATION CLIENT SAI field and the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header to the SAI values the device server is maintaining for the IKEv2-SCSI CCS, if any, being maintained for the I_T_L nexus on which the SECURITY PROTOCOL OUT command was received as described in 5.3.4; and

b) the device server shall decrypt and check the integrity of the Encrypted payload as described in 5.3.5.11.4.

Errors detected during the decryption and integrity checking of the Encrypted payload shall be handled as described in 5.3.8.2.

In the SECURITY PROTOCOL OUT command parameter list, the application client:

a) sends information about the SA usage and cryptographic algorithms;
b) sends its identity in the Identification – Application Client payload;
c) sends information proving knowledge of the secret corresponding to the application client's identity in the Authentication payload; and
d) integrity protects the Key Exchange step and the Authentication step using the Authentication payload.

The application client may include the Notify payload to send an initial contact notification to the device server. If sent, the initial contact notification specifies that the application client has no stored state for any SAs with the device server other than the SA that is being created.

In response to receipt of an initial contact notification, the device server should delete all other SAs that were authenticated with a SECURITY PROTOCOL OUT command that contained the same Identification - Application Client payload data as the payload data in the SECURITY PROTOCOL OUT command that the device server is processing.

If the device server deletes other SAs in response to an initial contact notification, the device server shall do so only after the successful completion of the Authentication step SECURITY PROTOCOL OUT command. If an error occurs during the Authentication SECURITY PROTOCOL OUT command, the device server shall ignore the initial contact notification.

If the device server is unable to proceed with SA creation for any reason (e.g., the verification of the Authentication payload fails), then the device server shall terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to ABORTED COMMAND, and the additional sense code set to an appropriate value. The additional sense code AUTHENTICATION FAILED shall be used if verification of the Authentication payload fails, or if authentication fails for any other reason.

4.1.3.7.3 Authentication step SECURITY PROTOCOL IN command

If the Authentication step SECURITY PROTOCOL OUT command (see 4.1.3.7.2) completes with GOOD status, then the application client sends a SECURITY PROTOCOL IN command (see SPC-5) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0103h to obtain the device server's authentication message. The parameter data consists of the IKEv2-SCSI header (see 5.3.4) and an Encrypted payload (see 5.3.5.11) that:

a) is integrity checked and encrypted in one of the following ways:
   A) using separate algorithms as follows:
      a) integrity checked using:
         A) the algorithm specified by the INTEG IKEv2-SCSI algorithm descriptor (see 5.3.6.4) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) from the Key Exchange step (see 4.1.3.6); and
         B) the SK_ar shared key (see 4.1.3.4);
         and
      b) encrypted using:
         A) the algorithm specified by the ENCR IKEv2-SCSI algorithm descriptor (see 5.3.6.2) in the IKEv2-SCSI SA Cryptographic Algorithms payload from the Key Exchange step; and

B) the SK_er shared key (see 4.1.3.4);

or

B) using a combined integrity check and encryption algorithm that uses the following (i.e., if the INTEG IKEv2-SCSI algorithm descriptor indicates AUTH_COMBINED):
   a) the algorithm specified by the ENCR IKEv2-SCSI algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload from the Key Exchange step; and
   b) the SK_er shared key with additional salt bytes as described in 4.1.3.8.1 and table 11 (see 4.1.3.4);

and

b) contains the following:
   1) an Identification – Device Server payload (see 5.3.5.4);
   2) an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 5.3.5.14);
   3) zero or more Certificate payloads (see 4.1.3.3.3.4); and
   4) an Authentication payload (see 5.3.5.7).

In the SECURITY PROTOCOL IN parameter data, the device server shall:

a) confirm information about the SA usage and cryptographic algorithms;
b) send its identity in the Identification – Device Server payload;
c) authenticate its identity; and
d) protect the integrity of the prior step messages using the Authentication payload.

Before completing the SECURITY PROTOCOL IN command with GOOD status, the device server shall generate the SA as described in 4.1.3.9.

The application client should verify the Authentication payload as described in 5.3.5.7. The Certificate payload(s) are used as part of this verification for PKI-based authentication. If the Authentication payload is verified and no other error occurs, the application client should generate the SA as described in 4.1.3.9.

If the application client is unable to proceed with SA creation for any reason (e.g., the verification of the Authentication payload fails), then the application client should:

a) not use the SA for any additional activities; and
b) notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 4.1.3.10.

The application client should compare the fields in the IKEv2-SCSI SAUT Cryptographic Algorithms payload to the values sent in the Authentication step SECURITY PROTOCOL OUT command (see 4.1.3.7.2). If the application client detects differences in the contents of the payloads, the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 4.1.3.10.

4.1.3.8 Generating shared keys

4.1.3.8.1 Overview

If the Authentication step is skipped (see 4.1.3.1), then shared key generation is performed as described in 4.1.3.8.2 and is summarized as follows:

a) the shared keys for SA management (e.g., SA creation and deletion) are generated at the same time as the shared keys used by the created SA; and
b) all the shared keys are generated during completion of the Key Exchange step (see 4.1.3.6.4).

If the Authentication step is processed (i.e., not skipped), then shared key generation is performed as described in 4.1.3.8.3 and is summarized as follows:

a) the shared keys for SA management (e.g., SA creation and deletion) are generated during the completion of the Key Exchange step (see 4.1.3.6.4); and
b) the shared keys used by the created SA are generated during SA generation (see 4.1.3.9).

Regardless of when the SA management shared keys (e.g., used for SA creation and deletion) and shared keys (see 4.1.3.4) used by the created SA are generated, the organization of the shared keys depends on the type of encryption and integrity checking algorithm being used as follows:

a) if an encryption algorithm that requires separate integrity checking is used, then separate shared keys are generated for each algorithm; or
b) if an encryption algorithm that includes integrity checking is used (i.e., if the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.4) contains AUTH_COMBINED), then no shared keys are generated for the integrity checking algorithm but additional key material is generated to act as salt bytes (see table 64 in 5.3.6.2) for the combined mode encryption algorithm.

4.1.3.8.2 Generating shared keys when the Authentication step is skipped

If the Authentication step is skipped (see 4.1.3.1), then the shared keys for SA management are generated at the same time as the shared keys for use by the created SA.

As part of completing the Key Exchange step (see 4.1.3.6.4), the SA participants generate all necessary shared keys as follows:

1) generate SKEYSEED (see RFC 7296) as described in 4.1.3.8.4;
2) generate the shared keys used for SA management as described in 4.1.3.8.5; and
3) as part of generating the SA (see 4.1.3.9) (i.e., as part of completing the Key Exchange step as described in 4.1.3.6.4), generate the shared keys for use by the created SA as described in 4.1.3.8.6 and store them in the KEYMAT SA parameter.

4.1.3.8.3 Generating shared keys when the Authentication step is processed

If the Authentication step is not skipped (see 4.1.3.1), then:

1) the shared keys for SA management are generated during completion of the Key Exchange step (see 4.1.3.6.4) as follows:
   1) generate SKEYSEED (see RFC 7296) as described in 4.1.3.8.4; and
   2) generate the shared keys used for SA management as described in 4.1.3.8.5;
   and
2) the shared keys for use by the created SA are generated during SA generation (see 4.1.3.9), near the end of processing for the Authentication step (see 4.1.3.7) as described in 4.1.3.8.6 and are stored in the KEYMAT SA parameter.

4.1.3.8.4 Initializing shared key generation

4.1.3.8.4.1 Initializing for SA creation shared key generation

The SA parameters are initialized for the KDF function used to generate SA creation shared keys as follows:

1) generate the input to the PRF by performing the last steps of the key exchange algorithm selected by the ALGORITHM IDENTIFIER field in the D-H IKEv2-SCSI algorithm descriptor (see 5.3.6.5) in the

IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) using at least the contents of one of the following fields as inputs to those last steps:
   A) the KEY EXCHANGE DATA field in the Key Exchange payload (see 5.3.5.3) in the Key Exchange SECURITY PROTOCOL OUT command (see 4.1.3.6.2) parameter data; and
   B) the KEY EXCHANGE DATA field in the Key Exchange payload in the Key Exchange SECURITY PROTOCOL IN command (see 4.1.3.6.3) parameter data;
2) generate SKEYSEED (see RFC 7296) using the output from step 1) and the PRF selected by the ALGORITHM IDENTIFIER field in the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload;
3) store the generated SKEYSEED value in the KEY_SEED SA parameter;
4) store the contents of the IKE_SA APPLICATION CLIENT SAI field from the IKEv2-SCSI Header (see 5.3.4) from the Key Exchange step SECURITY PROTOCOL IN command (see 4.1.3.6.3) in the AC_SAI SA parameter;
5) store the contents of the IKE_SA DEVICE SERVER SAI field from the IKEv2-SCSI Header (see 5.3.4) from the Key Exchange step SECURITY PROTOCOL IN command (see 4.1.3.6.3) in the DS_SAI SA parameter;
6) store the contents of the NONCE DATA field from the Nonce payload (see 5.3.5.8) from the parameter data for the Key Exchange step SECURITY PROTOCOL OUT command (see 4.1.3.6.2) in the AC_NONCE SA parameter; and
7) store the contents of the NONCE DATA field from the Nonce payload (see 5.3.5.8) from the parameter data for the Key Exchange step SECURITY PROTOCOL IN command (see 4.1.3.6.3) in the DS_NONCE SA parameter.

4.1.3.8.4.2 Initializing for generation of shared keys used by the created SA

The SA parameters are initialized for the KDF function used to generate the shared keys that are used by the created SA as follows:

1) store the SK_d value that was generated along with the other shared keys used in SA creation (see 4.1.3.8.5) in the KEY_SEED SA parameter;
2) store the contents of the IKE_SA APPLICATION CLIENT SAI field from the IKEv2-SCSI Header (see 5.3.4) from the Key Exchange step SECURITY PROTOCOL IN command (see 4.1.3.6.3) in the AC_SAI SA parameter;
3) store the contents of the IKE_SA DEVICE SERVER SAI field from the IKEv2-SCSI Header (see 5.3.4) from the Key Exchange step SECURITY PROTOCOL IN command (see 4.1.3.6.3) in the DS_SAI SA parameter;
4) store the contents of the NONCE DATA field from the Nonce payload (see 5.3.5.8) from the parameter data for the Key Exchange step SECURITY PROTOCOL OUT command (see 4.1.3.6.2) in the AC_NONCE SA parameter; and
5) store the contents of the NONCE DATA field from the Nonce payload (see 5.3.5.8) from the parameter data for the Key Exchange step SECURITY PROTOCOL IN command (see 4.1.3.6.3) in the DS_NONCE SA parameter.

4.1.3.8.5 Generating shared keys used for SA management

The shared keys used for SA management (e.g., SA creation and deletion) are generated using the SKEYSEED generated during initialization (see 4.1.3.8.4) and the steps described in this subclause.

Correct operation of SA management requires that the application client and device server use the same value for each shared key. SA management operations attempted with different values for the same shared key result in errors (e.g., integrity check errors).

Which shared keys are generated for SA management depends on:

a) whether the encryption algorithm includes integrity checking as indicated by the contents of the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.4) of the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13); and
b) whether the Authentication step is skipped (see 4.1.3.1).

Using the contents of the initialized SA parameters (see 4.1.3.8.4.1), the INTEG ALGORITHM IDENTIFIER field, and the KDF selected by the ALGORITHM IDENTIFIER field in the PRF IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload the following shared keys (see 4.1.3.4) are generated in the order shown:

a) if the INTEG ALGORITHM IDENTIFIER field is not set to AUTH_COMBINED, then generate the following shared keys (see 4.1.3.4):
   A) if the Authentication step is not skipped, generate the following shared keys:
      1) SK_d;
      2) SK_ai;
      3) SK_ar;
      4) SK_ei;
      5) SK_er;
      6) SK_pi; and
      7) SK_pr;

      or
   B) if the Authentication step is skipped, generate the following shared keys:
      1) SK_d;
      2) SK_ai;
      3) SK_ar;
      4) SK_ei; and
      5) SK_er;

   or
b) if the INTEG ALGORITHM IDENTIFIER field is set to AUTH_COMBINED, then generate the following shared keys:
   A) if the Authentication step is not skipped, generate the following shared keys:
      1) SK_d;
      2) SK_ei with additional salt bytes as described in 4.1.3.8.1 and table 11 (see 4.1.3.4);
      3) SK_er with additional salt bytes as described in 4.1.3.8.1 and table 11 (see 4.1.3.4);
      4) SK_pi; and
      5) SK_pr;

      or
   B) if the Authentication step is skipped, generate the following shared keys:
      1) SK_d;
      2) SK_ei with additional salt bytes as described in 4.1.3.8.1 and table 11 (see 4.1.3.4); and
      3) SK_er with additional salt bytes as described in 4.1.3.8.1 and table 11 (see 4.1.3.4).

The shared keys thus generated are combined with other data and stored in the MGMT_DATA SA parameter as described in 4.1.3.9.

How to determine the sizes of the shared keys to be generated is summarized in table 11 (see 4.1.3.4).

4.1.3.8.6 Generating shared keys for use by the created SA

As part of completing the Authentication step and generating the SA (see 4.1.3.9), the SA participants initialize the SA parameters for performing a KDF (see 4.1.3.8.4.2), and use the KDF selected by the ALGORITHM IDENTIFIER field in the PRF IKEv2-SCSI cryptographic algorithm descriptor in the Key Exchange step

IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) to generate the following shared keys (see 4.1.3.4) in the order shown and store them in the KEYMAT SA parameter:

a) if the ALGORITHM IDENTIFIER field in the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.2) contains ENCR_NULL in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 5.3.5.14), then generate the following shared keys:
   1) SK_ai; and
   2) SK_ar;

   and
b) if the ALGORITHM IDENTIFIER field in the ENCR IKEv2-SCSI cryptographic algorithm descriptor does not contain ENCR_NULL in the IKEv2-SCSI SAUT Cryptographic Algorithms payload, then generate the following shared keys:
   A) if the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.4) does not contain AUTH_COMBINED in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 5.3.5.14), then generate the following shared keys:
      1) SK_ai;
      2) SK_ar;
      3) SK_ei; and
      4) SK_er;

   or
   B) if the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor contains AUTH_COMBINED in the IKEv2-SCSI SAUT Cryptographic Algorithms payload, then generate the following shared keys:
      1) SK_ei with additional salt bytes as described in 4.1.3.8.1 and table 11 (see 4.1.3.4); and
      2) SK_er with additional salt bytes as described in 4.1.3.8.1 and table 11 (see 4.1.3.4).

How to determine the sizes of the shared keys to be generated is summarized in table 11 (see 4.1.3.4).

4.1.3.9 IKEv2-SCSI SA generation

Depending on whether or not the Authentication step was skipped (see 4.1.3.1), the SA participants shall generate shared keys as described in 4.1.3.8.

The SA participants shall initialize the SA parameters as follows:

1) KEYMAT shall be set as follows:
   A) if the Authentication step is skipped, KEYMAT shall be set as described in 4.1.3.8.2; or
   B) if the Authentication step is processed, KEYMAT shall be set as described in 4.1.3.8.3;

   and
2) the other SA parameters shall be set as follows:
   A) AC_SAI shall be set to the value in the IKE_SA APPLICATION CLIENT SAI field in the IKEv2-SCSI header (see 5.3.4) in the Key Exchange step SECURITY PROTOCOL OUT command (see 4.1.3.6.2);
   B) DS_SAI shall be set to the value in the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header in the Key Exchange step SECURITY PROTOCOL IN command (see 4.1.3.6.3);
   C) TIMEOUT shall be set to the IKEV2-SCSI SA INACTIVITY TIMEOUT field in the IKEv2-SCSI Timeout Values payload (see 5.3.5.15) in the Key Exchange step SECURITY PROTOCOL OUT command;
   D) KDF_ID shall be set to the value in the ALGORITHM IDENTIFIER field in the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13);
   E) AC_SQN shall be set to one;
   F) DS_SQN shall be set to one;
   G) AC_NONCE shall be set to zero;

H) DS_NONCE shall be set to zero;
I) KEY_SEED shall be set to zero;
J) USAGE_TYPE shall be set to the value in the SA TYPE field in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 5.3.5.14) in the Key Exchange step SECURITY PROTOCOL OUT command;
K) USAGE_DATA shall contain at least the following values from the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 5.3.5.14) in either the Key Exchange step SECURITY PROTOCOL OUT command or the Authentication step SECURITY PROTOCOL OUT command:
   a) the ALGORITHM IDENTIFIER field and KEY LENGTH field from the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.2);
   b) the ALGORITHM IDENTIFIER field from the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.4); and
   c) the contents, if any, of the USAGE DATA field;

   and
L) MGMT_DATA shall contain at least the following values:
   a) from the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) in the Key Exchange step SECURITY PROTOCOL OUT command:
      A) the ALGORITHM IDENTIFIER field and KEY LENGTH field from the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.2); and
      B) the ALGORITHM IDENTIFIER field from the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.4);
   b) from the shared keys generated for SA management (see 4.1.3.8.5), the following shared keys and salt bytes:
      A) the value of SK_ai, if any;
      B) the value of SK_ar, if any;
      C) the value of SK_ei, with additional salt bytes, if any; and
      D) the value of SK_er, with additional salt bytes, if any;

      and
   c) the next value of the MESSAGE ID field in the IKEv2-SCSI header.

NOTE 6 - The inclusion of the algorithm identifiers and key length in MGMT_DATA SA parameter enables the SA to apply the same encryption and integrity algorithms that IKEv2-SCSI negotiated to future IKEv2-SCSI SECURITY PROTOCOL OUT commands and IKEv2-SCSI SECURITY PROTOCOL IN commands, if any.

4.1.3.10 Abandoning an IKEv2-SCSI CCS

The occurrence of errors in either the application client or the device server may require that an IKEv2-SCSI CCS be abandoned.

A device server shall indicate that it has abandoned an IKEv2-SCSI CCS, if any, by terminating an IKEv2-SCSI CCS command (see 4.1.3.1) received on the I_T_L nexus for which the IKEv2-SCSI CCS state is being maintained with any combination of status and sense data other than those shown in table 13.

Table 13 — IKEv2-SCSI command terminations that do not abandon the CCS

IKEv2-SCSI CCS command	Status (Sense Key)	Additional Sense Code	Description
SECURITY PROTOCOL OUT SECURITY PROTOCOL IN	GOOD (n/a)	n/a	Indicates IKEv2-SCSI CCS is progressing as described in this standard
Key Exchange step SECURITY PROTOCOL OUT (see 4.1.3.6.2)	CHECK CONDITION (ABORTED COMMAND)	CONFLICTING SA CREATION REQUEST	At least one IKEv2-SCSI CCS is already active, and attempts to start another are blocked until the first CCS completes
SECURITY PROTOCOL OUT SECURITY PROTOCOL IN	CHECK CONDITION (NOT READY)	LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS	Device server is busy processing another command in the IKEv2-SCSI CCS associated with this I_T_L nexus or a different IKEv2-SCSI CCS associated with a different I_T_L nexus
SECURITY PROTOCOL IN	CHECK CONDITION (ILLEGAL REQUEST)	INVALID FIELD IN CDB	Incorrect SECURITY PROTOCOL IN CDB format
Authentication step SECURITY PROTOCOL OUT (see 4.1.3.7.2)		UNABLE TO DECRYPT PARAMETER LIST	Device server is unable to decrypt the Encrypted payload (see 5.3.5.11) or the integrity check fails
SECURITY PROTOCOL OUT		SA CREATION PARAMETER VALUE REJECTED	Indicates device server detection of an error that may be the result of another application client attempting to perform a denial of service attack against the application client that initiated the CCS (see 5.3.8.2)

As part of abandoning an IKEv2-SCSI CCS, the device server shall:

a) discard all maintained state (see 4.1.3.1); and
b) prepare to allow a future Key Exchange step SECURITY PROTOCOL OUT command received on any I_T_L nexus to start a new IKEv2-SCSI CCS.

After a device server abandons an IKEv2-SCSI CCS, the device server shall respond to all new IKEv2-SCSI protocol commands as if an IKEv2-SCSI CCS had never been started.

An application client should not abandon an IKEv2-SCSI CCS when the next command in the CCS is a SECURITY PROTOCOL IN command. Instead, the application client should send the appropriate SECURITY PROTOCOL IN command and then abandon the IKEv2-SCSI CCS.

An application client should specify that it has abandoned an IKEv2-SCSI CCS by sending an IKEv2-SCSI Delete operation (see 4.1.3.11) with application client SAI and device server SAI information that matches that of the IKEv2-SCSI CCS being abandoned.

4.1.3.11 Deleting an IKEv2-SCSI SA

As part of deleting an SA, both sets of SA parameters (see 4.1.1.2) are deleted as follows:

1) the application client uses the information in its SA parameters to prepare an IKEv2-SCSI Delete operation that requests deletion of the device server's SA parameters;
2) the application client deletes its SA parameters and any associated data;
3) the application client sends the IKEv2-SCSI Delete operation prepared in step 1) to the device server; and
4) in response to the IKEv2-SCSI Delete operation, the device server deletes its SA parameters and any associated data.

The IKEv2-SCSI Delete operation is a SECURITY PROTOCOL OUT command with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0104h. The parameter data consists of the IKEv2-SCSI header (see 5.3.4) and an Encrypted payload (see 5.3.5.11) that contains one Delete payload (see 5.3.5.10).

The Delete payload should conform to the requirements described in 5.3.5.10.

The device server shall process a valid Delete operation SECURITY PROTOCOL OUT command regardless of whether or not IKEv2-SCSI CCS state is being maintained for the I_T_L nexus on which the command is received.

The application client SAI and device server SAI information in a valid Delete operation may not identify an IKEv2-SCSI CCS for which the device server is maintaining state for the I_T_L nexus on which the command is received (see 5.3.5.11). This shall not be considered an error.

4.1.4 Security progress indication

The cryptographic calculations required by some security protocols are capable of consuming significant amounts of time in the device server. While cryptographic security calculations are in progress, the device server shall provide pollable REQUEST SENSE data (see SPC-5) with:

a) the sense key set to NOT READY;
b) the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
c) the PROGRESS INDICATION field set to indicate the progress of the device server in performing the necessary cryptographic calculations.

The device server shall not use the progress indication to report the detailed progress of cryptographic computations that take a variable amount of time based on their inputs. The device server may use the progress indication to report synthetic progress that does not reveal the detailed progress of the computation (e.g., divide a constant expected time for the computation by 10 and advance the progress indication in 10 % increments based on the elapsed time as compared to the expected time).

The requirements in this subclause apply to implementations of Diffie-Hellman computations and operations involving any keys (e.g., RSA) that optimize operations on large numbers based on the values of inputs (e.g., a computational step may be skipped if a bit or set of bits in an input is zero). A progress indication that advances based on the computation structure (e.g., count of computational steps) may reveal the time taken by content-dependent portions of the computation, and reveal information about the inputs.

4.1.5 ESP-SCSI encapsulations for parameter data

4.1.5.1 Overview

Subclause 4.1.5 defines a method for transferring encrypted and/or integrity checked parameter data in Data-In Buffers, Data-Out Buffers, variable length CDBs (see SPC-5), and Extended CDBs (see SPC-5). This method is based on the Encapsulating Security Payload (see RFC 4303) standard developed by the IETF.

NOTE 7 - Because of the constrained usage of ESP-SCSI parameter data in Data-In Buffers and/or Data-Out Buffers, the method defined in this standard differs from the one found in RFC 4303.

4.1.5.2 ESP-SCSI required inputs

Prior to using the ESP-SCSI descriptors defined in 4.1.5, an SA shall be created (see 4.1.1.3) with SA parameters that conform to the requirements defined in 4.1.1.2 and to the following:

a) the USAGE_TYPE SA parameter shall be set to a value for which ESP-SCSI usage is defined in table 4 (see 4.1.1.2);
b) the USAGE_DATA SA parameter shall contain at least the following:
   A) the algorithm identifier and key length for the encryption algorithm (e.g., the ALGORITHM IDENTIFIER field and KEY LENGTH field from the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.2) in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 5.3.5.14) negotiated by an IKEv2-SCSI SA creation protocol (see 4.1.3)); and
   B) the algorithm identifier for the integrity algorithm (e.g., the ALGORITHM IDENTIFIER field from the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.4) in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 5.3.5.14) negotiated by an IKEv2-SCSI SA creation protocol (see 4.1.3));
   and
c) the KEYMAT SA parameter shall consist of the shared keys described in 4.1.3.8.6.

ESP-SCSI uses the following additional information derived from the contents of the USAGE_DATA SA parameter:

a) the encryption algorithm identifier shall indicate:
   A) the absence of encryption by being set to ENCR_NULL (see table 25 in 4.1.6);
   B) the size of the initialization vector, if any (e.g., as shown in table 64 (see 5.3.6.2));
   C) the size of the salt bytes, if any (e.g., as shown in table 64 (see 5.3.6.2)); and
   D) for combined mode encryption algorithms, the size of the integrity check value (i.e., the algorithm's MAC length as shown in table 64 (see 5.3.6.2));
   and
b) the integrity algorithm identifier shall indicate:
   A) the use of a combined mode encryption algorithm being set to AUTH_COMBINED (see table 25 in 4.1.6); and
   B) for non-combined mode encryption algorithms, the size of the integrity check value (e.g., as shown in table 68 (see 5.3.6.4)).

Each shared key in KEYMAT shall be derived from the KDF generated bits in the order shown in 4.1.3.8.6. The size of each of the shared keys in KEYMAT is determined by the negotiated encryption algorithm and integrity algorithm as described in 4.1.3.4.

4.1.5.3 ESP-SCSI data format before encryption and after decryption

Before data bytes are encrypted and after they are decrypted, they have the format shown in table 14.

Table 14 — ESP-SCSI data format before encryption and after decryption

Bit / Byte	7	6	5	4	3	2	1	0
0	UNENCRYPTED BYTES							
•••								
p-1								
p	PADDING BYTES							
•••								
j-1								
j	PAD LENGTH (j-p)							
j+1	MUST BE ZERO							

The UNENCRYPTED BYTES field contains the bytes that are to be protected via encryption or that have been decrypted.

Before encryption, the PADDING BYTES field contains zero to 255 bytes. The number of padding bytes is:

a) defined by the encryption algorithm; or
b) the number required to cause the length of all bytes prior to encryption (i.e., j+2) to be a whole multiple of the alignment (see table 64 in 5.3.6.2) for the encryption algorithm being used.

The contents of the padding bytes are:

a) defined by the encryption algorithm; or
b) if the encryption algorithm does not define the padding bytes contents, a series of one byte binary values starting at one and incrementing by one in each successive byte (i.e., 01h in the first padding byte, 02h in the second padding byte, etc.).

If the encryption algorithm does not place requirements on the contents of the padding bytes option (i.e., option b)), then after decryption the contents of the padding bytes shall be verified to match the series of one byte binary values described in this subclause. If this verification is not successful, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set to indicate the last byte in the encrypted data (see SPC-5). If this verification is not successful in an application client, the decrypted data should be ignored.

The PAD LENGTH field is set to the number of bytes in the PADDING BYTES field.

The MUST BE ZERO field is set to zero. After decryption, the contents of the MUST BE ZERO field shall be verified to be zero. If this verification is not successful, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set to indicate the last byte

in the encrypted data (see SPC-5). If this verification is not successful in an application client, the decrypted data should be ignored.

4.1.5.4 ESP-SCSI outbound data descriptors

4.1.5.4.1 Overview

If ESP-SCSI is used in a variable length CDB (see SPC-5), an Extended CDB (see SPC-5), or parameter list data that appears in a Data-Out Buffer, then the parameter list data contains one or more descriptors selected based on the criteria shown in table 15.

Table 15 — ESP-SCSI outbound data descriptors

Descriptor name	External descriptor length [a]	Initialization vector present [b]	Reference
ESP-SCSI CDB	No	No	4.1.5.4.2.1
	No	Yes	4.1.5.4.2.2
ESP-SCSI Data-Out Buffer	No	No	4.1.5.4.2.1
	No	Yes	4.1.5.4.2.2
ESP-SCSI Data-Out Buffer without length	Yes	No	4.1.5.4.3.1
	Yes	Yes	4.1.5.4.3.2

[a] Yes means the data format defined for the Data-Out Buffer parameter data includes a length for the ESP-SCSI descriptor.

[b] See the USAGE_DATA SA parameter description in 4.1.5.2 for more information about the initialization vector.

4.1.5.4.2 ESP-SCSI CDBs or Data-Out Buffer parameter lists including a descriptor length

4.1.5.4.2.1 Initialization vector absent

If the USAGE_DATA SA parameter (see 4.1.5.2) indicates an encryption algorithm whose initialization vector size is zero, then the variable length CDB (see SPC-5), Extended CDB (see SPC-5), or Data-Out Buffer parameter list descriptor shown in table 16 contains the ESP-SCSI data.

Table 16 — ESP-SCSI CDBs or Data-Out Buffer parameter list descriptor without initialization vector

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				DESCRIPTOR LENGTH (n-1)				(LSB)
2								
3				Reserved				
4	(MSB)							
•••				DS_SAI				
7								(LSB)
8	(MSB)							
•••				DS_SQN				
15								(LSB)
16								
•••				ENCRYPTED OR AUTHENTICATED DATA				
i-1								
i	(MSB)							
•••				INTEGRITY CHECK VALUE				
n								(LSB)

The DESCRIPTOR LENGTH field specifies the number of bytes that follow in the ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter list descriptor.

The DS_SAI field is set to the value in the DS_SAI SA parameter (see 4.1.1.2) for the SA that is being used to prepare the ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter list descriptor. If the DS_SAI value is not known to the device server, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set (see SPC-5).

The DS_SQN field should be set to one plus the value in the application client's DS_SQN SA parameter (see 4.1.1.2) for the SA that is being used to prepare the ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter list descriptor. Before sending the ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter list, the application client should copy the contents of the DS_SQN field to its DS_SQN SA parameter.

The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set (see SPC-5) if any of the following conditions occur:

a) the DS_SQN field is set to zero;

b) the value in the DS_SQN field is less than or equal to the value in the device server's DS_SQN SA parameter; or
c) the value in the DS_SQN field is greater than 32 plus the value in the device server's DS_SQN SA parameter.

If the DS_SQN SA parameter is equal to FFFF FFFF FFFF FFFFh, the device server shall delete the SA.

The INITIALIZATION VECTOR field, if any, contains a value that is used as an input into the encryption algorithm and/or integrity algorithm specified by the SA specified by the DS_SAI field. The INITIALIZATION VECTOR field is not encrypted. The encryption algorithm and/or integrity algorithm may define additional requirements for the INITIALIZATION VECTOR field.

The ENCRYPTED OR AUTHENTICATED DATA field contains:

a) if an encryption algorithm for the SA specified by the DS_SAI field is not ENCR_NULL, encrypted data bytes for the following:
   1) the bytes in the UNENCRYPTED BYTES field (see 4.1.5.3);
   2) the bytes in the PADDING BYTES field (see 4.1.5.3);
   3) the byte that is the PAD LENGTH field (see 4.1.5.3); and
   4) the byte that is the MUST BE ZERO field (see 4.1.5.3);

   or
b) otherwise, the unencrypted data bytes.

If the integrity algorithm for the SA specified by the DS_SAI field is AUTH_COMBINED (see 4.1.5.2), then the AAD input to the encryption algorithm is composed of the following concatenated bytes:

1) the bytes in the DS_SAI field; and
2) the bytes in the DS_SQN field.

The INTEGRITY CHECK VALUE field contains a value that is computed as follows:

a) if the integrity algorithm is not AUTH_COMBINED, the integrity check value is computed using the specified integrity algorithm with the following concatenated bytes as inputs:
   1) the bytes in the DS_SAI field;
   2) the bytes in the DS_SQN field;
   3) the bytes in the INITIALIZATION VECTOR field, if any; and
   4) the bytes in the ENCRYPTED OR AUTHENTICATED DATA field after encryption, if any, has been performed;

   or
b) if the integrity algorithm is AUTH_COMBINED, the integrity check value is computed as an additional output of the specified encryption algorithm.

Upon receipt of ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter data, the device server shall compute an integrity check value for the ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter data as specified by the algorithms specified by the SA specified by the DS_SAI field using the inputs shown in this subclause. If the computed integrity check value does not match the value in the INTEGRITY CHECK VALUE field, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set (see SPC-5).

If the command is not terminated due to a sequence number error or a mismatch between the computed integrity check value and the contents of the INTEGRITY CHECK VALUE field, then the device server shall copy the contents of the received DS_SQN field to its DS_SQN SA parameter.

4.1.5.4.2.2 Initialization vector present

If the USAGE_DATA SA parameter indicates an encryption algorithm whose initialization vector size (i.e., s) is greater than zero, then the variable length CDB (see SPC-5), Extended CDB (see SPC-5), or Data-Out Buffer parameter data descriptor shown in table 17 contains the ESP-SCSI data.

Table 17 — ESP-SCSI CDBs or Data-Out Buffer full parameter list descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				DESCRIPTOR LENGTH (n-1)				(LSB)
2								
3				Reserved				
4	(MSB)							
•••				DS_SAI				
7								(LSB)
8	(MSB)							
•••				DS_SQN				
15								(LSB)
16	(MSB)							
•••				INITIALIZATION VECTOR				
16+s-1								(LSB)
16+s								
•••				ENCRYPTED OR AUTHENTICATED DATA				
i-1								
i	(MSB)							
•••				INTEGRITY CHECK VALUE				
n								(LSB)

The DESCRIPTOR LENGTH field, DS_SAI field, DS_SQN field, INITIALIZATION VECTOR field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 4.1.5.4.2.1.

4.1.5.4.3 ESP-SCSI Data-Out Buffer parameter lists for externally specified descriptor length

4.1.5.4.3.1 Initialization vector absent

If the USAGE_DATA SA parameter (see 4.1.5.2) indicates an encryption algorithm whose initialization vector size is zero and the length of the ESP-SCSI Data-Out Buffer parameter list descriptor is specified in the same parameter list that contains the descriptor, then the Data-Out Buffer parameter list descriptor shown in table 18 contains the ESP-SCSI data.

Table 18 — ESP-SCSI Data-Out Buffer parameter list descriptor without length and initialization vector

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
3								
4	(MSB)							
•••	DS_SAI							
7								(LSB)
8	(MSB)							
•••	DS_SQN							
15								(LSB)
16	ENCRYPTED OR AUTHENTICATED DATA							
•••								
i-1								
i	(MSB)							
•••	INTEGRITY CHECK VALUE							
n								(LSB)

The DS_SAI field, DS_SQN field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 4.1.5.4.2.1.

4.1.5.4.3.2 Initialization vector present

If the USAGE_DATA SA parameter indicates an encryption algorithm whose initialization vector size (i.e., s) is greater than zero and the length of the ESP-SCSI Data-Out Buffer parameter list descriptor is specified in the same parameter list that contains the descriptor, then the Data-Out Buffer parameter list descriptor shown in table 19 contains the ESP-SCSI data.

Table 19 — ESP-SCSI Data-Out Buffer parameter list descriptor without length

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
3								
4	(MSB)							
•••	DS_SAI							
7								(LSB)
8	(MSB)							
•••	DS_SQN							
15								(LSB)
16	(MSB)							
•••	INITIALIZATION VECTOR							
16+s-1								(LSB)
16+s	ENCRYPTED OR AUTHENTICATED DATA							
•••								
i-1								
i	(MSB)							
•••	INTEGRITY CHECK VALUE							
n								(LSB)

The DS_SAI field, DS_SQN field, INITIALIZATION VECTOR field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 4.1.5.4.2.1.

4.1.5.5 ESP-SCSI Data-In Buffer parameter data descriptors

4.1.5.5.1 Overview

A device server shall transfer ESP-SCSI parameter data descriptors in a Data-In Buffer only in response to a request that specifies an SA using the AC_SAI SA parameter and DS_SAI SA parameter values (see 4.1.1.2). If the specified combination of AC_SAI and DS_SAI values in a command that requests the transfer of ESP-SCSI parameter data descriptors is not known to the device server, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST or to INVALID FIELD IN CDB, the SKSV bit set to one, and SENSE KEY SPECIFIC field set (see SPC-5).

If ESP-SCSI is used in parameter data which appears in a Data-In Buffer, the parameter data contains one or more descriptors selected based on the criteria shown in table 20.

Table 20 — ESP-SCSI Data-In Buffer parameter data descriptors

Descriptor name	External descriptor length [a]	Initialization vector present [b]	Reference
ESP-SCSI Data-In Buffer	No	No	4.1.5.5.2.1
	No	Yes	4.1.5.5.2.2
ESP-SCSI Data-In Buffer without length	Yes	No	4.1.5.5.3.1
	Yes	Yes	4.1.5.5.3.2

[a] Yes means the data format defined for the Data-In Buffer parameter data includes a length for the ESP-SCSI descriptor.

[b] See the USAGE_DATA SA parameter description in 4.1.5.2 for more information about the initialization vector.

If ESP-SCSI parameter data descriptors are used in a Data-In Buffer, then the outbound data (see 4.1.5.4) should include at least one ESP-SCSI descriptor using the same SA.

4.1.5.5.2 ESP-SCSI Data-In Buffer parameter data including a descriptor length

4.1.5.5.2.1 Initialization vector absent

If the USAGE_DATA SA parameter (see 4.1.5.2) indicates an encryption algorithm whose initialization vector size is zero, then the Data-In Buffer parameter data descriptor shown in table 21 contains the ESP-SCSI data.

Table 21 — ESP-SCSI Data-In Buffer parameter data descriptor without initialization vector

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				DESCRIPTOR LENGTH (n-1)				(LSB)
2								
3				Reserved				
4	(MSB)							
•••				AC_SAI				
7								(LSB)
8	(MSB)							
•••				AC_SQN				
15								(LSB)
16								
•••				ENCRYPTED OR AUTHENTICATED DATA				
i-1								
i	(MSB)							
•••				INTEGRITY CHECK VALUE				
n								(LSB)

The DESCRIPTOR LENGTH field specifies the number of bytes that follow in the ESP-SCSI Data-In Buffer parameter data descriptor.

The AC_SAI field is set to the value in the AC_SAI SA parameter (see 4.1.1.2) for the SA that is being used to prepare the ESP-SCSI Data-In Buffer parameter data descriptor. If the AC_SAI value is not known to the application client, the ESP-SCSI data-in parameter data descriptor should be ignored.

The AC_SQN field is set to one plus the value in the device server's AC_SQN SA parameter (see 4.1.1.2) for the SA that is being used to prepare the ESP-SCSI data-on buffer parameter data descriptor. Before sending the ESP-SCSI Data-Out Buffer parameter list as part of a command that completes with GOOD status, the device server shall copy the contents of the AC_SQN field to its AC_SQN SA parameter. The device server shall not send two ESP-SCSI Data-Out Buffer parameter data descriptors that contain the same values in AC_SAI field and AC_SQN field.

If the AC_SQN SA parameter is equal to FFFF FFFF FFFF FFFFh, the device server shall delete the SA after the Data-In Buffer parameter data containing that value is sent.

The application client should ignore the ESP-SCSI data-in parameter data descriptor if any of the following occur:

a) the AC_SQN field is set to zero;
b) the value in the AC_SQN field is less than or equal to the value in the application client's AC_SQN SA parameter; or
c) the value in the AC_SQN field is greater than 32 plus the value in the application client's AC_SQN SA parameter.

The INITIALIZATION VECTOR field, if any, contains a value that is used as an input into the encryption algorithm and/or integrity algorithm specified by the SA specified by the AC_SAI field. The INITIALIZATION VECTOR field is not encrypted. The encryption algorithm and/or integrity algorithm may define additional requirements for the INITIALIZATION VECTOR field.

The ENCRYPTED OR AUTHENTICATED DATA field contains:

a) if an encryption algorithm for the SA specified by the AC_SAI field is not ENCR_NULL, encrypted data bytes for the following:
    1) the bytes in the UNENCRYPTED BYTES field (see 4.1.5.3);
    2) the bytes in the PADDING BYTES field (see 4.1.5.3);
    3) the byte that is the PAD LENGTH field (see 4.1.5.3); and
    4) the byte that is the MUST BE ZERO field (see 4.1.5.3);

    or
b) otherwise, the unencrypted data bytes.

If the integrity algorithm for the SA specified by the AC_SAI field is AUTH_COMBINED (see 4.1.5.2), then the AAD input to the encryption algorithm is composed of the following concatenated bytes:

1) the bytes in the AC_SAI field; and
2) the bytes in the AC_SQN field;

The INTEGRITY CHECK VALUE field contains a value that is computed as follows:

a) if the integrity algorithm is not AUTH_COMBINED, the integrity check value is computed using the specified integrity algorithm with the following concatenated bytes as inputs:
    1) the bytes in the AC_SAI field;
    2) the bytes in the AC_SQN field;

3) the bytes in the INITIALIZATION VECTOR field, if any; and
4) the bytes in the ENCRYPTED OR AUTHENTICATED DATA field after encryption, if any, has been performed;

or

b) if the integrity algorithms is AUTH_COMBINED, the integrity check value is computed as an additional output of the specified encryption algorithm.

Upon receipt of ESP-SCSI Data-In Buffer parameter data, the application client should compute an integrity check value for the ESP-SCSI parameter data as specified by the algorithms specified by the SA specified by the AC_SAI field using the inputs shown in this subclause. If the computed integrity check value does not match the value in the INTEGRITY CHECK VALUE field, the results returned by the command should be ignored.

The application client should copy the contents of the AC_SQN field to its AC_SQN SA parameter if all of the following occur:

a) the command completed with GOOD status;
b) the ESP-SCSI data-in parameter data descriptor was not ignored due to inconsistency problems with the AC_SQN field; and
c) the computed integrity check value matched the contents of the INTEGRITY CHECK VALUE field.

4.1.5.5.2.2 Initialization vector present

If the USAGE_DATA SA parameter indicates an encryption algorithm whose initialization vector size (i.e., s) is greater than zero, then the Data-In Buffer parameter data descriptor shown in table 22 contains the ESP-SCSI data.

Table 22 — ESP-SCSI Data-In Buffer full parameter data descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				DESCRIPTOR LENGTH (n-1)				(LSB)
2								
3				Reserved				
4	(MSB)							
•••				AC_SAI				
7								(LSB)
8	(MSB)							
•••				AC_SQN				
15								(LSB)
16	(MSB)							
•••				INITIALIZATION VECTOR				
16+s-1								(LSB)
16+s								
•••				ENCRYPTED OR AUTHENTICATED DATA				
i-1								
i	(MSB)							
•••				INTEGRITY CHECK VALUE				
n								(LSB)

The DESCRIPTOR LENGTH field, AC_SAI field, AC_SQN field, INITIALIZATION VECTOR field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 4.1.5.5.2.1.

4.1.5.5.3 ESP-SCSI Data-In Buffer parameter data for externally specified descriptor length

4.1.5.5.3.1 Initialization vector absent

If the USAGE_DATA SA parameter (see 4.1.5.2) indicates an encryption algorithm whose initialization vector size is zero and the length of the ESP-SCSI Data-In Buffer parameter data descriptor is specified in the same parameter data that contains the descriptor, then the Data-In Buffer parameter data descriptor shown in table 23 contains the ESP-SCSI data.

Table 23 — ESP-SCSI Data-In Buffer parameter data descriptor without length and initialization vector

Bit / Byte	7	6	5	4	3	2	1	0
0								
•••				Reserved				
3								
4	(MSB)							
•••				AC_SAI				
7								(LSB)
8	(MSB)							
•••				AC_SQN				
15								(LSB)
16								
•••				ENCRYPTED OR AUTHENTICATED DATA				
i-1								
i	(MSB)							
•••				INTEGRITY CHECK VALUE				
n								(LSB)

The AC_SAI field, AC_SQN field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 4.1.5.5.2.1.

4.1.5.5.3.2 Initialization vector present

If the USAGE_DATA SA parameter indicates an encryption algorithm whose initialization vector size (i.e., s) is greater than zero and the length of the ESP-SCSI Data-In Buffer parameter data descriptor is specified in the same parameter data that contains the descriptor, then the Data-In Buffer parameter data descriptor shown in table 24 contains the ESP-SCSI data.

Table 24 — ESP-SCSI Data-In Buffer parameter data descriptor without length

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
3								
4	(MSB)			AC_SAI				
•••								
7								(LSB)
8	(MSB)			AC_SQN				
•••								
15								(LSB)
16	(MSB)			INITIALIZATION VECTOR				
•••								
16+s-1								(LSB)
16+s	ENCRYPTED OR AUTHENTICATED DATA							
•••								
i-1								
i	(MSB)			INTEGRITY CHECK VALUE				
•••								
n								(LSB)

The AC_SAI field, AC_SQN field, INITIALIZATION VECTOR field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 4.1.5.5.2.1.

4.1.6 Security algorithm codes

Table 25 lists the security algorithm codes used in security protocol parameter data.

Table 25 — Security algorithm codes (part 1 of 2)

Code	Description	Reference
Encryption algorithms		
0001 000Ch	CBC-AES-256-HMAC-SHA-1	IEEE 1619.1
0001 0010h	CCM-128-AES-256	IEEE 1619.1
0001 0014h	GCM-128-AES-256	IEEE 1619.1
0001 0016h	XTS-AES-256-HMAC-SHA-512	IEEE 1619.1
8001 000Bh [a]	ENCR_NULL	5.3.6.2
8001 000Ch [a]	AES-CBC	RFC 3602
8001 0010h [a]	AES-CCM with a 16-byte MAC	RFC 4309
8001 0014h [a]	AES-GCM with a 16-byte MAC	RFC 4106
8001 0400h to 8001 FFFFh	Vendor specific	
PRF and KDF algorithms [b]		
8002 0002h [a]	IKEv2-use based on SHA-1	table 66 (see 5.3.6.3)
8002 0004h [a]	IKEv2-use based on AES-128 in CBC mode	
8002 0005h [a]	IKEv2-use based on SHA-256	
8002 0007h [a]	IKEv2-use based on SHA-512	
8002 0400h to 8002 FFFFh	Vendor specific	
Integrity checking (i.e., AUTH) algorithms		
8003 0002h [a]	AUTH_HMAC_SHA1_96	RFC 2404
8003 000Ch [a]	AUTH_HMAC_SHA2_256_128	RFC 4868
8003 000Eh [a]	AUTH_HMAC_SHA2_512_256	RFC 4868
F003 0000h	AUTH_COMBINED	5.3.6.4
8003 0400h to 8003 FFFFh	Vendor specific	

[a] The lower order 16 bits of this code value are assigned to match an IANA assigned value, if any, for an equivalent IKEv2 encryption algorithm and values of 800xh in the high order 16 bits have x selected to match the IANA assigned IKEv2 transform type (e.g., 8001h – Encryption Algorithms, 8002h – PRFs and KDFs).

[b] PRFs are equivalent to the prf() functions defined in RFC 7296. KDFs are equivalent to the prf+() functions defined in RFC 7296.

[c] The low order 8 bits of this code value are assigned to match the AUTH METHOD field in the Authentication payload (see 5.3.5.7).

Table 25 — Security algorithm codes (part 2 of 2)

Code	Description	Reference
Diffie-Hellman algorithms		
8004 000Eh [a]	2 048-bit MODP group (finite field D-H)	RFC 3526
8004 000Fh [a]	3 072-bit MODP group (finite field D-H)	RFC 3526
8004 0010h [a]	4 096-bit MODP group (finite field D-H)	RFC 3526
8004 0011h [a]	6 144-bit MODP group (finite field D-H)	RFC 3526
8004 0012h [a]	8 192-bit MODP group (finite field D-H)	RFC 3526
8004 0013h [a]	256-bit random ECP group	RFC 5903
8004 0015h [a]	521-bit random ECP group	RFC 5903
8004 0400h to 8004 FFFFh	Vendor specific	
SA Authentication payload authentication algorithms		
00F9 0000h [c]	SA_AUTH_NONE	4.1.3.3.4 and 5.3.6.6
00F9 0001h [c]	RSA Digital Signature with SHA-1	RFC 7296
00F9 0002h [c]	Shared Key Message Integrity Code	RFC 7296
00F9 0009h [c]	ECDSA with SHA-256 on the P-256 curve	RFC 4754
00F9 000Bh [c]	ECDSA with SHA-512 on the P-521 curve	RFC 4754
00F9 00C9h to 00F9 00FFh	Vendor specific	
Other algorithms		
0000 0000h to 0000 FFFFh	Restricted	IANA
All others	Reserved	

[a] The lower order 16 bits of this code value are assigned to match an IANA assigned value, if any, for an equivalent IKEv2 encryption algorithm and values of 800xh in the high order 16 bits have x selected to match the IANA assigned IKEv2 transform type (e.g., 8001h – Encryption Algorithms, 8002h – PRFs and KDFs).

[b] PRFs are equivalent to the prf() functions defined in RFC 7296. KDFs are equivalent to the prf+() functions defined in RFC 7296.

[c] The low order 8 bits of this code value are assigned to match the AUTH METHOD field in the Authentication payload (see 5.3.5.7).

4.2 Secure random numbers

Secure random numbers should be generated as specified by RFC 4086 (e.g., see NIST SP 800-90 A).

If the same random number source is used to generate random numbers for multiple purposes (e.g., nonces and secret keys), then interactions between the two shall not be allowed to compromise secrecy. If the value sequence generated by the common random number source is predictable to any degree, then the random number values that are transmitted outside the SCSI device may provide information about the random number values that the SCSI device maintains internally, based on the reasonable assumption that an adversary knows the order in which the random numbers are obtained from the common random number source. SCSI devices shall eliminate sources of such predictability.

Compliance with RFC 4086 is one method for achieving the required independence between random number values.

5 Security protocol parameters for all device types

5.1 Security protocol information description

5.1.1 Overview

The security protocol information security protocol (i.e., the SECURITY PROTOCOL field set to 00h in a SECURITY PROTOCOL IN command) returns security protocol related information. A SECURITY PROTOCOL IN command with the SECURITY PROTOCOL field set to 00h is not associated with a previous SECURITY PROTOCOL OUT command and shall be processed without regard for whether a SECURITY PROTOCOL OUT command has been processed.

If the SECURITY PROTOCOL IN command is supported, the SECURITY PROTOCOL field set to 00h shall be supported as defined in this standard.

5.1.2 CDB description

If the SECURITY PROTOCOL field is set to 00h in a SECURITY PROTOCOL IN command, the contents of the SECURITY PROTOCOL SPECIFIC field are defined in table 26.

Table 26 — SECURITY PROTOCOL SPECIFIC field for SECURITY PROTOCOL IN protocol 00h

Code	Description	Support	Reference
0000h	Supported security protocol list	Mandatory	5.1.3
0001h	Certificate data	Mandatory	5.1.4
0002h	Security compliance information	Optional	5.1.5
All others	Reserved		

All other CDB fields for SECURITY PROTOCOL IN command shall meet the requirements stated in SPC-5.

Each time a SECURITY PROTOCOL IN command with the SECURITY PROTOCOL field set to 00h is received, the device server shall transfer the data defined in 5.1 starting with byte 0.

5.1.3 Supported security protocols list description

If the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL SPECIFIC field is set to 0000h in a SECURITY PROTOCOL IN command, then the parameter data shall have the format shown in table 27.

Table 27 — Supported security protocols SECURITY PROTOCOL IN parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
5								
6	(MSB)			SUPPORTED SECURITY PROTOCOL LIST LENGTH (m-7)				
7								(LSB)
Supported security protocol list								
8	SUPPORTED SECURITY PROTOCOL (00h) [first]							
	⋮							
m	SUPPORTED SECURITY PROTOCOL [last]							
m+1	Pad bytes (if any)							
•••								
n								

The SUPPORTED SECURITY PROTOCOL LIST LENGTH field indicates the total length, in bytes, of the supported security protocol list that follows.

Each SUPPORTED SECURITY PROTOCOL field in the supported security protocols list shall contain one of the security protocol values (see SPC-5) supported by the logical unit. The values shall be listed in ascending order starting with 00h.

The total data length shall conform to the ALLOCATION LENGTH field requirements (see SPC-5). Pad bytes may be appended to meet this length. Pad bytes shall have a value of 00h.

5.1.4 Certificate data description

5.1.4.1 Certificate overview

A certificate is either an X.509 Public Key Certificate (see 5.1.4.2) or an X.509 Attribute Certificate (see 5.1.4.3) depending on the capabilities of the logical unit.

If the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL SPECIFIC field is set to 0001h in a SECURITY PROTOCOL IN command, then the parameter data shall have the format shown in table 28.

Table 28 — Certificate data SECURITY PROTOCOL IN parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	CERTIFICATE LENGTH (m-3)						
3								(LSB)
4	CERTIFICATE							
•••								
m								
m+1	Pad bytes (if any)							
•••								
n								

The CERTIFICATE LENGTH field indicates the total length, in bytes, of the certificate or certificates that follow. The length may include more than one certificate. If the device server does not have a certificate to transfer, the CERTIFICATE LENGTH field shall be set to 0000h.

The contents of the CERTIFICATE field are defined in 5.1.4.2 and 5.1.4.3.

The total data length shall conform to the ALLOCATION LENGTH field requirements (see SPC-5). Pad bytes may be appended to meet this length. Pad bytes shall have a value of 00h.

5.1.4.2 Public Key certificate description

RFC 5280 and RFC 6818 define the certificate syntax for certificates consistent with X.509v3 Public Key Certificate Specification.

5.1.4.3 Attribute certificate description

RFC 5755 defines the certificate syntax for certificates consistent with X.509v2 Attribute Certificate Specification.

5.1.5 Security compliance information description

5.1.5.1 Security compliance information overview

The security compliance information parameter data contains information about security standards that apply to this SCSI target device.

If the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL SPECIFIC field is set to 0002h in a SECURITY PROTOCOL IN command, then the parameter data shall have the format shown in table 29.

Table 29 — Security compliance information SECURITY PROTOCOL IN parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••	SECURITY COMPLIANCE INFORMATION LENGTH (m-3)							
3								(LSB)
Compliance descriptors								
4								
•••	Compliance descriptor [first]							
	⋮							
•••	Compliance descriptor [last]							
m								
m+1								
•••	Pad bytes (if any)							
n								

The SECURITY COMPLIANCE INFORMATION LENGTH field indicates the total length, in bytes, of the compliance descriptors that follow.

Each compliance descriptor (see 5.1.5.2) contains information about a security standard that applies to this SCSI target device. Compliance descriptors may be returned in any order.

The total data length shall conform to the ALLOCATION LENGTH field requirements (see SPC-5). Pad bytes may be appended to meet this length. Pad bytes shall have a value of 00h.

5.1.5.2 Compliance descriptor overview

The format of a compliance descriptor in the security compliance information SECURITY PROTOCOL IN parameter data is shown in table 30.

Table 30 — Compliance descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				COMPLIANCE DESCRIPTOR TYPE				(LSB)
2								
3				Reserved				
4	(MSB)							
•••				COMPLIANCE DESCRIPTOR LENGTH (n-3)				
7								(LSB)
8								
•••				Descriptor specific information				
n								

The COMPLIANCE DESCRIPTOR TYPE field (see table 31) indicates the format of the descriptor specific information. The security compliance information SECURITY PROTOCOL IN parameter data may contain more than one compliance descriptor with the same value in the COMPLIANCE DESCRIPTOR TYPE field.

Table 31 — COMPLIANCE DESCRIPTOR TYPE field

Code	Description	Related standards	Reference
0001h	Security requirements for cryptographic modules	FIPS 140-2 FIPS 140-3	5.1.5.3
All others	Reserved		

The COMPLIANCE DESCRIPTOR LENGTH field indicates the number of bytes that follow in the compliance descriptor.

The contents of the descriptor specific information depend on the value in the COMPLIANCE DESCRIPTOR TYPE field.

5.1.5.3 FIPS 140 compliance descriptor

The FIPS 140 compliance descriptor (see table 32) contains information that may be used to locate information about a FIPS 140 certificate associated with the SCSI target device. The SCSI target device may or may not be operating in the mode specified by that certificate.

Table 32 — FIPS 140 compliance descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				COMPLIANCE DESCRIPTOR TYPE (0001h)				(LSB)
2								
3				Reserved				
4	(MSB)							
•••				COMPLIANCE DESCRIPTOR LENGTH (0000 0208h)				
7								(LSB)
8	RELATED STANDARD							
9	OVERALL SECURITY LEVEL							
10								
•••				Reserved				
15								
16	(MSB)							
•••				COMPLIANCE DESCRIPTOR HARDWARE VERSION				
143								(LSB)
144	(MSB)							
•••				COMPLIANCE DESCRIPTOR VERSION				
271								(LSB)
272	(MSB)							
•••				COMPLIANCE DESCRIPTOR MODULE NAME				
527								(LSB)

The COMPLIANCE DESCRIPTOR TYPE field and COMPLIANCE DESCRIPTOR LENGTH field are defined in 5.1.5.2 and shall be set as shown in table 32 for the FIPS 140 compliance descriptor.

The RELATED STANDARD field (see table 33) is an ASCII data field (see SPC-5) that indicates the related standard described by this compliance descriptor.

Table 33 — RELATED STANDARD field

Code	Related standard
00h	none specified
32h	FIPS 140-2
33h	FIPS 140-3
All others	Reserved

The OVERALL SECURITY LEVEL field is an ASCII data field (see SPC-5) that indicates the FIPS 140 overall security level that is reported by NIST or CSEC.

The COMPLIANCE DESCRIPTOR HARDWARE VERSION field is null-terminated, null-padded data (see SPC-5) that indicates the version number of the hardware in the module, as reported by NIST or CSEC.

The COMPLIANCE DESCRIPTOR VERSION field is null-terminated, null-padded data (see SPC-5) that indicates the version number of the firmware or software in the module, as reported by NIST or CSEC. The value in the COMPLIANCE DESCRIPTOR VERSION field is not related to the PRODUCT REVISION LEVEL field of standard INQUIRY data (see SPC-5).

The COMPLIANCE DESCRIPTOR MODULE NAME field is null-terminated, null-padded data (see SPC-5) that indicates the name or identifier of the cryptographic module, as reported by NIST or CSEC.

5.2 SA creation capabilities

5.2.1 Overview

If the SECURITY PROTOCOL field in a SECURITY PROTOCOL IN command (see SPC-5) is set to 40h, then the command returns information related to the SA creation (see 4.1.1.3) capabilities provided by the device server.

The SA creation capabilities protocol is independent of any other SA creation protocols. If any SA creation protocols are supported (e.g., IKEv2-SCSI (see 4.1.3)), then the device server shall not refuse to process an SA creation capabilities SECURITY PROTOCOL IN command, and this processing shall not affect the state maintained for any SA creation CCS (e.g., an IKEv2-SCSI CCS) on any I_T_L nexus. Except for those cases where an SA creation capabilities SECURITY PROTOCOL IN command reports changed SA creation capabilities, processing of the command shall not affect the concurrent processing of any commands that are part of an SA creation CCS.

If any SA creation protocols are supported, the SA creation capabilities protocol shall be supported as described in 5.2.

The SA creation capabilities SECURITY PROTOCOL IN CDB requirements are described in 5.2.2.

As shown in table 34 (see 5.2.2), the format of the parameter data for a SA creation capabilities SECURITY PROTOCOL IN command depends on the value in the SECURITY PROTOCOL SPECIFIC field in the CDB.

5.2.2 SA creation capabilities CDB description

The SA creation capabilities SECURITY PROTOCOL IN CDB has the format defined in SPC-5 with the additional requirements described in this subclause.

If the SECURITY PROTOCOL field is set to SA creation capabilities (i.e., 40h) in a SECURITY PROTOCOL IN command, then the SECURITY PROTOCOL SPECIFIC field (see table 34) identifies the SA creation protocol (see 4.1.1.3) for which the device server shall return capability information.

Table 34 — SECURITY PROTOCOL SPECIFIC field for the SA creation capabilities

Code	Description	Parameter data format
0000h	Supported device server capabilities formats	5.2.3.1
0001h to 0100h	Reserved	
0101h	IKEv2-SCSI device server capabilities	5.2.3.2
0102h to EFFFh	Reserved	
F000h to FFFFh	Vendor Specific	

If an SA creation capabilities SECURITY PROTOCOL IN command is received with the INC_512 bit set to one, then the device server shall terminate the SECURITY PROTOCOL IN command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

5.2.3 SA creation capabilities parameter data formats

5.2.3.1 Supported device server capabilities formats parameter data format

The supported device server capabilities formats parameter data (see table 35) indicates the capabilities parameter data formats that the device server supports.

Table 35 — Supported device server capabilities formats parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••	PARAMETER DATA LENGTH (n-3)							
3								(LSB)
Supported capabilities parameter data formats								
4	(MSB)							
5	CAPABILITIES PARAMETER DATA FORMAT (0000h) [first]							(LSB)
	⋮							
n-1	(MSB)							
n	CAPABILITIES PARAMETER DATA FORMAT [last]							(LSB)

The PARAMETER DATA LENGTH field indicates the number of bytes that follow in the parameter data.

Each CAPABILITIES PARAMETER DATA FORMAT field in the supported capabilities parameter data formats list shall contain one of the SECURITY PROTOCOL SPECIFIC field values (see table 34) supported by the device server. The values shall be listed in ascending order starting with 0000h.

5.2.3.2 IKEv2-SCSI device server capabilities parameter data format

The IKEv2-SCSI device server capabilities parameter data (see table 36) indicates the IKEv2 transforms (i.e., key exchange protocols and authentication protocols) supported by the device server for IKEv2-SCSI.

Table 36 — IKEv2-SCSI device server capabilities parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PARAMETER DATA LENGTH (n-3)				
3								(LSB)
4								
•••				IKEv2-SCSI SA Creation Capabilities payload (see 5.3.5.12)				
n								

The PARAMETER DATA LENGTH field indicates the number of bytes that follow in the parameter data.

The IKEv2-SCSI SA Creation Capabilities payload (see 5.3.5.12) indicates the algorithms supported by IKEv2-SCSI in the Key Exchange step (see 4.1.3.6) and Authentication step (see 4.1.3.7).

The primary content of the IKEv2-SCSI device server capabilities SA creation capabilities parameter data is an IKEv2-SCSI payload (see 5.3.5) that is used by the device server and application client in the construction of other IKEv2-SCSI payloads (see 4.1.3).

5.3 IKEv2-SCSI

5.3.1 Overview

If the SECURITY PROTOCOL field in a SECURITY PROTOCOL OUT command (see SPC-5) or a SECURITY PROTOCOL IN command (see SPC-5) is set to 41h, then the command is part of an IKEv2-SCSI CCS (see 4.1.3) and is used to transfer IKEv2-SCSI protocol information to or from the device server.

In an IKEv2-SCSI CCS, a defined sequence of SECURITY PROTOCOL OUT commands and SECURITY PROTOCOL IN commands are sent by the application client and processed by the device server as summarized in 4.1.3.1.

The IKEv2-SCSI SECURITY PROTOCOL OUT CDB format is described in 5.3.3.

The IKEv2-SCSI SECURITY PROTOCOL IN CDB format is described in 5.3.2.

The IKEv2-SCSI SECURITY PROTOCOL OUT command and the IKEv2-SCSI SECURITY PROTOCOL IN command use the same parameter data format (see 5.3.4). The primary content of the IKEv2-SCSI parameter data is one or more IKE payloads (see 5.3.5).

If the IKEv2-SCSI SA creation protocol is supported (see 5.1), the SA creation capabilities protocol (see 5.2) shall also be supported.

5.3.2 IKEv2-SCSI SECURITY PROTOCOL IN CDB description

The IKEv2-SCSI SECURITY PROTOCOL IN CDB has the format defined in SPC-5 with the additional requirements described in this subclause.

If the SECURITY PROTOCOL field is set to IKEv2-SCSI (i.e., 41h) in a SECURITY PROTOCOL IN command, the SECURITY PROTOCOL SPECIFIC field (see table 37) identifies the IKEv2-SCSI step (see 4.1.3.1) that the device server is to process. If the IKEv2-SCSI SA creation protocol is supported (see 5.1), the SECURITY PROTOCOL IN command support requirements are shown in table 37.

Table 37 — SECURITY PROTOCOL SPECIFIC field as defined by the IKEv2-SCSI SECURITY PROTOCOL IN command

Code	Description	Support	References	
			Usage	Data format
0000h to 00FFh	Restricted		RFC 7296	RFC 7296
0100h to 0101h	Reserved			
0102h	Key Exchange step	Mandatory	4.1.3.6.3	5.3.4
0103h	Authentication step	Mandatory	4.1.3.7.3	5.3.4
0104h to EFFFh	Reserved			
F000h to FFFFh	Vendor Specific			

If an IKEv2-SCSI SECURITY PROTOCOL IN command is received with the INC_512 bit set to one while the device server is maintaining state for an IKEv2-SCSI CCS on the I_T_L nexus on which the command was received (see 4.1.3.1), then:

a) the device server shall terminate the SECURITY PROTOCOL IN command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
b) the device server shall continue the IKEv2-SCSI CCS by preparing to receive another SECURITY PROTOCOL IN command or SECURITY PROTOCOL OUT command.

If an IKEv2-SCSI SECURITY PROTOCOL IN command is received with the INC_512 bit set to one while the device server is not maintaining state for an IKEv2-SCSI CCS (see 4.1.3.1), then the device server shall terminate the SECURITY PROTOCOL IN command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

5.3.3 IKEv2-SCSI SECURITY PROTOCOL OUT CDB description

The IKEv2-SCSI SECURITY PROTOCOL OUT CDB has the format defined in SPC-5 with the additional requirements described in this subclause.

If the SECURITY PROTOCOL field is set to IKEv2-SCSI (i.e., 41h) in a SECURITY PROTOCOL OUT command, the SECURITY PROTOCOL SPECIFIC field (see table 38) identifies the IKEv2-SCSI step (see 4.1.3.1) that the device server is to process. If the IKEv2-SCSI SA creation protocol is supported (see 5.1), the SECURITY PROTOCOL OUT command support requirements are shown in table 38.

Table 38 — SECURITY PROTOCOL SPECIFIC field as defined by the IKEv2-SCSI SECURITY PROTOCOL OUT command

Code	Description	Support	References	
			Usage	Data format
0000h to 00FFh	Restricted		RFC 7296	RFC 7296
0100h to 0101h	Reserved			
0102h	Key Exchange step	Mandatory	4.1.3.6.3	5.3.4
0103h	Authentication step	Mandatory	4.1.3.7.3	5.3.4
0104h	Delete operation	Mandatory	4.1.3.11	5.3.4
0105h to EFFFh	Reserved			
F000h to FFFFh	Vendor Specific			

If an IKEv2-SCSI SECURITY PROTOCOL OUT command is received with the INC_512 bit set to one while the device server is maintaining state for an IKEv2-SCSI CCS on the I_T_L nexus on which the command was received (see 4.1.3.1), then:

a) the device server shall terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
b) the device server shall continue the IKEv2-SCSI CCS by preparing to receive another SECURITY PROTOCOL OUT command or SECURITY PROTOCOL IN command.

If an IKEv2-SCSI SECURITY PROTOCOL OUT command is received with the INC_512 bit set to one while the device server is not maintaining state for an IKEv2-SCSI CCS (see 4.1.3.1), then the device server shall terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

Any IKEv2-SCSI SECURITY PROTOCOL OUT command with a transfer length of up to 16 384 bytes shall not be terminated with an error due to the number of bytes to be transferred and processed.

5.3.4 IKEv2-SCSI parameter data format

Table 39 shows the parameter list format used by a SECURITY PROTOCOL OUT command and the parameter data format used by a SECURITY PROTOCOL IN command if the SECURITY PROTOCOL field is set to IKEv2-SCSI (i.e., 41h).

Table 39 — IKEv2-SCSI SECURITY PROTOCOL OUT command and SECURITY PROTOCOL IN command parameter data

Bit / Byte	7	6	5	4	3	2	1	0
IKEv2-SCSI header								
0	Restricted (see RFC 7296)							
•••								
3								
4	(MSB) IKE_SA APPLICATION CLIENT SAI							
•••								
7								(LSB)
8	Restricted (see RFC 7296)							
•••								
11								
12	(MSB) IKE_SA DEVICE SERVER SAI							
•••								
15								(LSB)
16	NEXT PAYLOAD							
17	MAJOR VERSION (2h)				MINOR VERSION			
18	EXCHANGE TYPE							
19	Reserved			INTTR	VERSION	RSPNS	Reserved	
20	(MSB) MESSAGE ID							
•••								
23								(LSB)
24	(MSB) IKE LENGTH (n+1)							
•••								
27								(LSB)
IKEv2-SCSI payloads								
28	IKEv2-SCSI payload (see 5.3.5) [first]							
•••								
	⋮							
	IKEv2-SCSI payload (see 5.3.5) [last]							
•••								
n								

The IKE_SA APPLICATION CLIENT SAI field specifies the value that is or is destined to become the AC_SAI SA parameter (see 4.1.1.2) in the generated SA (see 4.1.3.9). The AC_SAI is chosen by the application client to uniquely identify its representation of the SA that is being negotiated or managed (e.g., deleted).

If the device server receives an IKEv2-SCSI header with the IKE_SA APPLICATION CLIENT SAI field set to zero, then the error shall be processed as described in 5.3.8.

The application client should compare the IKE_SA APPLICATION CLIENT SAI field contents in any SECURITY PROTOCOL IN parameter data to the value that the application client is maintaining for the IKEv2-SCSI CCS or SA management. If the two values are not identical, the application client should abandon the IKEv2-SCSI CCS, if any, and notify the device server that the IKEv2-SCSI CCS, if any, is being abandoned as described in 4.1.3.10.

Except in the Key Exchange step SECURITY PROTOCOL OUT command (see 4.1.3.6.2), the IKE_SA DEVICE SERVER SAI field specifies the value that is or is destined to become the DS_SAI SA parameter in the generated SA. The DS_SAI is chosen by the device server in accordance with the requirements in 4.1.1.1 to uniquely identify its representation of the SA that is being negotiated. In the Key Exchange step SECURITY PROTOCOL OUT command the IKE_SA DEVICE SERVER SAI field is reserved.

The application client should compare the IKE_SA DEVICE SERVER SAI field contents in the Authentication step SECURITY PROTOCOL IN parameter data to the value that the application client is maintaining for the IKEv2-SCSI CCS. If the two values are not identical, the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 4.1.3.10.

The device server shall validate the contents of the IKE_SA APPLICATION CLIENT SAI field and the IKE_SA DEVICE SERVER SAI field as shown in table 40.

Table 40 — IKEv2-SCSI header checking of SAIs

Contents of SECURITY PROTOCOL SPECIFIC field in SECURITY PROTOCOL OUT CDB	Expected contents for …		Device server action if expected field contents not received
	IKE_SA APPLICATION CLIENT SAI field	IKE_SA DEVICE SERVER SAI field	
0102h (i.e., Key Exchange step)	any value	reserved	No actions taken based on expected field contents.
0103h (i.e., Authentication step)	A match with the SAI values maintained for an IKEv2-SCSI CCS on the I_T_L nexus on which the command was received		The device server shall: a) terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE REJECTED; and b) continue the IKEv2-SCSI CCS by preparing to receive another Authentication step SECURITY PROTOCOL OUT command.
0104h (i.e., Delete operation)	If at least one IKEv2-SCSI CCS is being maintained for the I_T_L nexus on which the command was received, then a match with the SAI values maintained for: a) an IKEv2-SCSI CCS; or b) any active SA		The device server shall: a) terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE REJECTED; and b) continue the IKEv2-SCSI CCS by preparing to receive another Authentication step SECURITY PROTOCOL OUT command.
	If no IKEv2-SCSI CCS is being maintained for the I_T_L nexus on which the command was received, then a match with the SAI values maintained for any active SA		The device server shall terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The NEXT PAYLOAD field (see table 41) identifies the first IKEv2-SCSI payload that follows the IKEv2-SCSI header.

Table 41 — NEXT PAYLOAD field

Code	IKE Payload Name	Support requirements in SECURITY PROTOCOL …		Reference
		IN	OUT	
00h	No Next Payload	Mandatory		5.3.5.2
01h to 20h		Reserved		
21h	Security Association	Prohibited [a]		RFC 7296
22h	Key Exchange	Mandatory		5.3.5.3
23h	Identification – Application Client	Prohibited	Mandatory	5.3.5.4
24h	Identification – Device Server	Mandatory	Prohibited	5.3.5.4
25h	Certificate	Optional		5.3.5.5
26h	Certificate Request	Optional		5.3.5.6
27h	Authentication	Mandatory		5.3.5.7
28h	Nonce	Mandatory		5.3.5.8
29h	Notify	Prohibited	Mandatory	5.3.5.9
2Ah	Delete	Prohibited	Mandatory	5.3.5.10
2Bh	Vendor ID	Prohibited		RFC 7296
2Ch	Traffic Selector – Application Client	Prohibited		RFC 7296
2Dh	Traffic Selector – Device Server	Prohibited		RFC 7296
2Eh	Encrypted	Mandatory		5.3.5.11
2Fh	Configuration	Prohibited		RFC 7296
30h	Extensible Authentication	Prohibited		RFC 7296
31h to 7Fh		Restricted		RFC 7296
80h	IKEv2-SCSI SA Creation Capabilities	Mandatory		5.3.5.12
81h	IKEv2-SCSI SA Cryptographic Algorithms	Mandatory		5.3.5.13
82h	IKEv2-SCSI SAUT Cryptographic Algorithms	Mandatory		5.3.5.14
83h	IKEv2-SCSI Timeout Values	Mandatory		5.3.5.15
84h to BFh		Reserved		
C0h to FFh	Vendor Specific			

[a] The Security Association payload type value is not used in IKEv2-SCSI. The IKEv2-SCSI SA Cryptographic Algorithms payload (i.e., 81h) and IKEv2-SCSI SAUT Cryptographic Algorithms payload (i.e., 82h) are used instead.

The MAJOR VERSION field shall contain the value 2h. If a device server receives an IKEv2-SCSI header with a MAJOR VERSION field containing a value other than 2h, then the error shall be processed as described in 5.3.8.

The MINOR VERSION field is reserved.

The EXCHANGE TYPE field is reserved.

The initiator (INTTR) bit shall be set to:

a) one for SECURITY PROTOCOL OUT commands; and
b) zero for SECURITY PROTOCOL IN commands.

If a device server receives an IKEv2-SCSI header with the INTTR bit set to zero, the error shall be processed as described in 5.3.8.

If an application client receives an IKEv2-SCSI header with the INTTR bit set to one, the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 4.1.3.10.

The VERSION bit is reserved.

The response (RSPNS) bit shall be set to:

a) zero for SECURITY PROTOCOL OUT commands; and
b) one for SECURITY PROTOCOL IN commands.

If a device server receives an IKEv2-SCSI header with the RSPNS bit set to one, the error shall be processed as described in 5.3.8.

If an application client receives an IKEv2-SCSI header with the RSPNS bit set to zero, the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 4.1.3.10.

The MESSAGE ID field (see table 42) identifies the function of the parameter data.

Table 42 — MESSAGE ID field

Code	Description
0000 0000h	Key Exchange step SECURITY PROTOCOL OUT command (see 4.1.3.6.2) or SECURITY PROTOCOL IN command (see 4.1.3.6.3)
0000 0001h	Authentication step SECURITY PROTOCOL OUT command (see 4.1.3.7.2) or SECURITY PROTOCOL IN command (see 4.1.3.7.3)
0000 0002h	Delete operation in a SECURITY PROTOCOL OUT command
All others	Reserved

If the device server receives a SECURITY PROTOCOL OUT command with an invalid MESSAGE ID field in its IKEv2-SCSI header, then the error shall be processed as described in 5.3.8.

If the application client receives an invalid MESSAGE ID field in the parameter data for a SECURITY PROTOCOL IN command, then the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 4.1.3.10.

The IKE LENGTH field specifies the total number of bytes in the parameter data, including the IKEv2-SCSI header and all the IKEv2-SCSI payloads.

NOTE 8 - The contents of the IKE LENGTH field differ from those found in most SCSI length fields. However, they are consistent with the IKEv2 usage (see RFC 7296).

Each IKEv2-SCSI payload (see 5.3.5) contains specific data related to the operation being performed. A specific combination of IKEv2-SCSI payloads is specified for each operation (e.g., Key Exchange) as summarized in 4.1.3.2. The Encryption payload (see 5.3.6.2) nests one set of IKEv2-SCSI payloads inside another.

Based on the contents of the SECURITY PROTOCOL SPECIFIC field in the SECURITY PROTOCOL OUT CDB, the device server shall:

a) validate the contents of the NEXT PAYLOAD field in the IKEv2-SCSI header as shown in table 43, before performing any decryption or integrity checking; and
b) validate that the number of instances of each next payload value shown in table 43 occur in the parameter data after the encrypted data, if any, is decrypted and integrity checked (i.e., if the NEXT PAYLOAD field in the IKEv2-SCSI header contains 2Eh, after the Encrypted payload is decrypted and integrity checked).

If in the parameter data for a SECURITY PROTOCOL OUT command the next payload values in the IKEv2-SCSI header and in the unencrypted payloads in the IKEv2-SCSI payloads (see table 39) do not meet the requirements shown in table 43 for the listed SECURITY PROTOCOL SPECIFIC field contents, then the error shall be processed as described in 5.3.8. Otherwise, the device server shall not process an error due to an unexpected ordering of next payload values in the parameter data.

Based on the contents of the SECURITY PROTOCOL SPECIFIC field in the SECURITY PROTOCOL IN command, the device server shall:

a) place one of the next payload values allowed by table 43 in the NEXT PAYLOAD field in the IKEv2-SCSI header; and
b) include the number of instances of each next payload value shown in table 43 in the parameter data independent of whether the resulting payloads are encrypted or not.

Table 43 — Next payload values in SECURITY PROTOCOL OUT/IN parameter data (part 1 of 3)

Next payload value	Next payload value allowed in IKEv2-SCSI header	Number of times a next payload value is allowed
SECURITY PROTOCOL OUT command with SECURITY PROTOCOL SPECIFIC field set to 0102h (i.e., Key Exchange step)		**Authentication step not skipped (see 4.1.3.1)**
83h (i.e., IKEv2-SCSI Timeout Values)	Yes	1
81h (i.e., IKEv2-SCSI SA Cryptographic Algorithms)	Yes	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	No	0
22h (i.e., Key Exchange)	Yes	1
28h (i.e., Nonce)	Yes	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0

Table 43 — Next payload values in SECURITY PROTOCOL OUT/IN parameter data (part 2 of 3)

Next payload value	Next payload value allowed in IKEv2-SCSI header	Number of times a next payload value is allowed
SECURITY PROTOCOL OUT command with SECURITY PROTOCOL SPECIFIC field set to 0102h (i.e., Key Exchange step)		**Authentication step skipped (see 4.1.3.1)**
83h (i.e., IKEv2-SCSI Timeout Values)	Yes	1
81h (i.e., IKEv2-SCSI SA Cryptographic Algorithms)	Yes	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	Yes	1
22h (i.e., Key Exchange)	Yes	1
28h (i.e., Nonce)	Yes	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0
SECURITY PROTOCOL IN command with SECURITY PROTOCOL SPECIFIC field set to 0102h (i.e., Key Exchange step)		**Authentication step not skipped (see 4.1.3.1)**
81h (i.e., IKEv2-SCSI SA Cryptographic Algorithms)	Yes	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	No	0
22h (i.e., Key Exchange)	Yes	1
28h (i.e., Nonce)	Yes	1
26h (i.e., Certificate Request)	Yes	0 or more
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0
SECURITY PROTOCOL IN command with SECURITY PROTOCOL SPECIFIC field set to 0102h (i.e., Key Exchange step)		**Authentication step skipped (see 4.1.3.1)**
81h (i.e., IKEv2-SCSI SA Cryptographic Algorithms)	Yes	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	Yes	1
22h (i.e., Key Exchange)	Yes	1
28h (i.e., Nonce)	Yes	1
26h (i.e., Certificate Request)	Yes	0 or more
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0

Table 43 — Next payload values in SECURITY PROTOCOL OUT/IN parameter data (part 3 of 3)

Next payload value	Next payload value allowed in IKEv2-SCSI header	Number of times a next payload value is allowed
SECURITY PROTOCOL OUT command with SECURITY PROTOCOL SPECIFIC field set to 0103h (i.e., Authentication step)		
2Eh (i.e., Encrypted)	Yes	1
23h (i.e., Identification – Application Client)	No	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	No	1
25h (i.e., Certificate)	No	0 or more
26h (i.e., Certificate Request)	No	0 or more
29h (i.e., Notify)	No	0 or 1
27h (i.e., Authentication)	No	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0
SECURITY PROTOCOL IN command with SECURITY PROTOCOL SPECIFIC field set to 0103h (i.e., Authentication step)		
2Eh (i.e., Encrypted)	Yes	1
24h (i.e., Identification – Device Server)	No	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	No	1
25h (i.e., Certificate)	No	0 or more
27h (i.e., Authentication)	No	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0
SECURITY PROTOCOL OUT command with SECURITY PROTOCOL SPECIFIC field set to 0104h (i.e., Delete operation)		
2Eh (i.e., Encrypted)	Yes	1
2Ah (i.e., Delete)	No	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0

5.3.5 IKEv2-SCSI payloads

5.3.5.1 IKEv2-SCSI payload format

Each IKEv2-SCSI payload (see table 44) is composed of a header and data that is specific to the payload type.

Table 44 — IKEv2-SCSI payload format

Bit / Byte	7	6	5	4	3	2	1	0
IKEv2-SCSI payload header								
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
IKEv2-SCSI payload-specific data								
4	Payload-specific data							
•••								
n								

The NEXT PAYLOAD field identifies the IKEv2-SCSI payload that follows this IKEv2-SCSI payload using one of the code values shown in table 41 (see 5.3.4).

The device server shall set the critical (CRIT) bit to one in all IKEv2-SCSI payloads returned to the application client. The application client should set the CRIT bit to one in all IKEv2-SCSI payloads sent to the device server.

If a device server receives an IKEv2-SCSI payload that it does not recognize (e.g., an IKEv2-SCSI payload identified by a next payload value of 01h) with the CRIT bit set to one, then the error shall be processed as described in 5.3.8.

If an application client receives an IKEv2-SCSI payload that it does not recognize with the CRIT bit set to one, then the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 4.1.3.10.

If an application client or device server receives an IKEv2-SCSI payload that it does not recognize with the CRIT bit set to zero, then it should use the NEXT PAYLOAD field and the IKE PAYLOAD LENGTH field to skip processing of the unrecognized IKEv2-SCSI payload and continue processing at the next IKEv2-SCSI payload.

The IKE PAYLOAD LENGTH field specifies the total number of bytes in the payload, including the IKEv2-SCSI payload header. The value in the IKE PAYLOAD LENGTH field is not required to be a multiple of two or four (i.e., no byte alignment is maintained among IKEv2-SCSI payloads).

NOTE 9 - The contents of the IKE PAYLOAD LENGTH field differ from those found in most SCSI length fields. However, they are consistent with the IKEv2 usage (see RFC 7296).

The format and contents of the IKEv2-SCSI payload-specific data depends on the value in the NEXT PAYLOAD field of:

a) the IKEv2-SCSI header (see 5.3.4), if this is the first IKEv2-SCSI payload in the parameter data; or
b) the previous IKEv2-SCSI payload, in all other cases.

5.3.5.2 No Next payload

A NEXT PAYLOAD field that is set to 00h (i.e., No Next payload) specifies that no more IKEv2-SCSI payloads follow the current payload. The IKEv2-SCSI No Next payload contains no bytes.

5.3.5.3 Key Exchange payload

The Key Exchange payload (see table 45) transfers Diffie-Hellman shared key exchange data between an application client and a device server or vice versa.

Table 45 — Key Exchange payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	(MSB)	DIFFIE-HELLMAN GROUP NUMBER						
5								(LSB)
6		Reserved						
7								
8		KEY EXCHANGE DATA						
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 5.3.5.1.

The DIFFIE-HELLMAN GROUP NUMBER field specifies the least significant 16 bits from ALGORITHM IDENTIFIER field in the D-H IKEv2-SCSI algorithm descriptor (see 5.3.6.5) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13).

The KEY EXCHANGE DATA field specifies the sender's Diffie-Hellman public value for this key exchange. The binary representation of key exchange data is defined in the reference cited for the Diffie-Hellman group that is used (see table 70).

When a prime modulus (i.e., mod p) Diffie-Hellman group is used, the length of the Diffie-Hellman public value shall be equal to the length of the prime modulus over which the exponentiation was performed as shown in table 70 (see 5.3.6.5). Bits that are set to zero shall be prepended to the KEY EXCHANGE DATA field if necessary.

Diffie-Hellman exponential reuse and reuse of analogous Diffie-Hellman public values for Diffie-Hellman mechanisms not based on exponentiation:

a) should not be performed; and

b) if performed, shall be constrained (e.g., requirements for discarding Diffie-Hellman information) as defined in RFC 7296.

If Diffie-Hellman exponentials and public values are reused in IKEv2-SCSI, the associated random nonces shall not be reused and the new random nonces should have a large number of bits of entropy (see RFC 7296).

5.3.5.4 Identification – Application Client payload and Identification – Device Server payload

The Identification – Application Client payload (see table 46) transfers identification information from the application client to the device server. The Identification – Device Server payload (see table 46) transfers identification information from the device server to the application client.

Table 46 — Identification payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	ID TYPE							
5	Reserved							
7								
8	IDENTIFICATION DATA							
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 5.3.5.1.

The ID TYPE field describes the contents of the IDENTIFICATION DATA field and shall contain one of the named values shown in table 47.

Table 47 — ID TYPE field

Code	Name [a]	Contents of the IDENTIFICATION DATA field
09h	ID_DER_ASN1_DN	The value of a certificate subject field (see RFC 4945, RFC 5280 and RFC 6818)
0Ah	ID_DER_ASN1_GN	The value of a name contained in a Subject Alternative Name (i.e., SubjectAltName) certificate extension (see RFC 4945, RFC 5280 and RFC 6818)
0Bh	ID_KEY_ID	Arbitrary identity data (e.g., initiator port names, target port names, and SCSI device names)
0Ch	ID_FC_NAME	FC-SP-2 certificates that certify a Fibre Channel name as an identity to be used (see RFC 4595 and FC-SP-2)
All others	Prohibited	

[a] See RFC 7296 and RFC 4595 for additional information about the associated identification data for these names.

The contents of the IDENTIFICATION DATA field depend on the value in the ID TYPE field.

If the Certificate payload is included in the parameter data, the identity in the Identification – Application Client payload or Identification – Device Server payload is not required to match anything in the Certificate payload (see RFC 7296). Based on information provided by a configuration method that is outside the scope of this standard, device servers and application clients are required to verify a match between:

a) the identity in an Identification payload; and
b) the subject name or subject alternative name in a Certificate payload that contains an X.509 Certificate (see 5.3.5.5).

If a device server receives an Identification – Application Client payload that does not conform to the requirements in RFC 7296 or the requirements in this subclause, then the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 5.3.8.3.

If an application client receives an Identification – Device Server payload that does not conform to the requirements in RFC 7296 or the requirements in this subclause, then the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 4.1.3.10.

5.3.5.5 Certificate payload

The Certificate payload (see table 48) delivers a requested identity authentication certificate. The protocol for using Certificate payloads is described in 4.1.3.3.3.4.

Table 48 — Certificate payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	CERTIFICATE ENCODING							
5	CERTIFICATE DATA							
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 5.3.5.1.

The CERTIFICATE ENCODING field describes the contents of the CERTIFICATE DATA field and shall contain one of the values shown in table 49.

Table 49 — CERTIFICATE ENCODING field

Code	Description	Reference
00h	Reserved	
01h to 03h	Prohibited	Annex B
04h	X.509 Certificate - Signature	RFC 7296
05h to 0Ah	Prohibited	Annex B
0Bh	Raw RSA Key	RFC 5996
0Ch to 0Dh	Prohibited	Annex B
0Eh to C8h	Restricted	IANA
C9h to FFh	Reserved	

The contents of the CERTIFICATE DATA field depend on the value in the CERTIFICATE ENCODING field.

The relationship between the Certificate payload and the Identification payload is described in 5.3.5.4.

Device servers that support certificates should support a mechanism outside the scope of this standard for replacing certificates and have the ability to store more than one certificate to facilitate such replacements.

5.3.5.6 Certificate Request payload

The Certificate Request payload (see table 50) allows an application client or device server to request the use of certificates as part of identity authentication and to name one or more trust anchors (see RFC 7296) for the certificate verification process. The Certificate payload (see table 48) delivers a requested identity authentication certificate. The protocol for using Certificate Request payloads is described in 4.1.3.3.3.4.

Table 50 — Certificate Request payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	CERTIFICATE ENCODING							
Certification authority list								
5	(MSB)	CERTIFICATION AUTHORITY [first]						
•••								
24								(LSB)
	⋮							
n-19	(MSB)	CERTIFICATION AUTHORITY [last]						
•••								
n								(LSB)

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 5.3.5.1.

The value in the CERTIFICATE ENCODING field (see table 49 in 5.3.5.5) indicates the type or format of certificate being requested. Multiple Certificate Request payloads may be included in the parameter data transferred by a single command. If the parameter data contains more than one Certificate Request payload, each Certificate Request payload should have a different value in the CERTIFICATE ENCODING field.

Each CERTIFICATION AUTHORITY field contains an indicator of a trusted authority for the certificate type indicated by the CERTIFICATE ENCODING field in this Certificate Request payload. The indicator is a SHA-1 hash of the public key of a trusted certification authority. The indicator is encoded as the SHA-1 hash of the Subject Public Key Info element from the trust anchor certificate (see RFC 5280 and RFC 6818).

Device servers that support certificates should support a mechanism outside the scope of this standard for replacing certification authority values, and shall have the ability to store one or more certification authority values to facilitate such replacements.

5.3.5.7 Authentication payload

The Authentication payload (see table 51) allows the application client and a device server to verify that the data transfers in their IKEv2-SCSI CCS have not be compromised by a man-in-the-middle attack (see A.4).

Table 51 — Authentication payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	AUTH METHOD							
5		Reserved						
7								
8		AUTHENTICATION DATA						
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 5.3.5.1.

The AUTH METHOD field indicates the authentication algorithm to be applied to this Authentication payload. The AUTH METHOD field contains the least significant eight bits of the ALGORITHM IDENTIFIER field in an SA_AUTH_OUT algorithm descriptor or an SA_AUTH_IN algorithm descriptor (see 5.3.6.6) from the Key Exchange step (see 4.1.3.6).

If the contents of the AUTH METHOD field in the parameter data for the Authentication step SECURITY PROTOCOL OUT command (see 4.1.3.7.2) do not match the least significant eight bits in the ALGORITHM IDENTIFIER field in the SA_AUTH_OUT algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) in the Key Exchange step, then the device server shall:

a) terminate the SECURITY PROTOCOL OUT command CHECK CONDITION status, with the sense key set to ABORTED COMMAND and the additional sense code set to AUTHENTICATION FAILED; and
b) abandon the IKEv2-SCSI CCS (see 4.1.3.10).

If the contents of the AUTH METHOD field in the parameter data for the Authentication step SECURITY PROTOCOL IN command (see 4.1.3.7.3) do not match the least significant eight bits in the ALGORITHM IDENTIFIER field in the SA_AUTH_IN algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload in the Key Exchange step, then the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 4.1.3.10.

In the Authentication step SECURITY PROTOCOL OUT command (see 4.1.3.7.2) parameter list, the AUTHENTICATION DATA field contains the result of applying the algorithm specified by the ALGORITHM IDENTIFIER field in the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) in the Key Exchange step (see 4.1.3.6) as described in table

72 (see 5.3.6.6) and this subclause to generate the following concatenation of bytes:

1) all the bytes in the Data-In Buffer returned by the Device Server Capabilities step (see 4.1.3.5) SECURITY PROTOCOL IN command;
2) all the bytes in the Data-Out Buffer sent by the Key Exchange step SECURITY PROTOCOL OUT command (see 4.1.3.6.2) in the same IKEv2-SCSI CCS for which GOOD status was returned;
3) all the bytes in the IKEv2-SCSI payload-specific data part (see 5.3.5.1) of the Nonce payload (see 5.3.5.8) that was received in the Key Exchange step SECURITY PROTOCOL IN command in the same IKEv2-SCSI CCS; and
4) all the bytes produced by applying the PRF selected by the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) to the following inputs:
    1) the SK_pi shared key (see 4.1.3.4); and
    2) all the bytes in the IKEv2-SCSI payload-specific data part (see 5.3.5.1) of the Identification – Application Client payload (see 5.3.5.4).

While processing the Authentication step SECURITY PROTOCOL OUT command, the device server shall verify the contents of the AUTHENTICATION DATA field by applying the algorithm specified by the ALGORITHM IDENTIFIER field in the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) in the Key Exchange step (see 4.1.3.6) as described in table 72 (see 5.3.6.6) and this subclause to generate the following concatenation of bytes:

1) all the bytes in the Data-In Buffer that the device server returned to any application client in response to the last received Device Server Capabilities step SECURITY PROTOCOL IN command;
2) all the bytes in the Data-Out Buffer received in the Key Exchange step SECURITY PROTOCOL OUT command (see 4.1.3.6.2) in the same IKEv2-SCSI CCS for which GOOD status was returned;
3) all the bytes in the IKEv2-SCSI payload-specific data part (see 5.3.5.1) of the Nonce payload (see 5.3.5.8) sent in the Key Exchange step SECURITY PROTOCOL IN command in the same IKEv2-SCSI CCS; and
4) all the bytes produced by applying the PRF selected by the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) to the following inputs:
    1) the SK_pi shared key (see 4.1.3.4); and
    2) all the bytes received in the IKEv2-SCSI payload-specific data part (see 5.3.5.1) of the Identification – Application Client payload (see 5.3.5.4) of the parameter list being processed.

If the verification of the contents of the AUTHENTICATION DATA field is not successful, then the device server shall:

a) terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to ABORTED COMMAND and the additional sense code set to AUTHENTICATION FAILED; and
b) abandon the IKEv2-SCSI CCS (see 4.1.3.10).

For the Authentication step SECURITY PROTOCOL IN command (see 4.1.3.7.3) parameter list, the device server shall compute the AUTHENTICATION DATA field contents by applying the algorithm specified by the ALGORITHM IDENTIFIER field in the SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) in the Key Exchange step (see 4.1.3.6) as described in table 72 (see 5.3.6.6) and this subclause to generate the following concatenation of bytes:

1) all the bytes in the Data-In Buffer that the device server returned to any application client in response to the last received Device Server Capabilities step SECURITY PROTOCOL IN command;

2) all the bytes in the Data-In Buffer sent by the most recent Key Exchange step SECURITY PROTOCOL IN command (see 4.1.3.6.3) in the same IKEv2-SCSI CCS for which GOOD status was returned;
3) all the bytes in the IKEv2-SCSI payload-specific data part (see 5.3.5.1) of the Nonce payload (see 5.3.5.8) received in the Key Exchange step SECURITY PROTOCOL OUT command in the same IKEv2-SCSI CCS; and
4) all the bytes produced by applying the PRF selected by the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) to the following inputs:
    1) the SK_pr shared key (see 4.1.3.4); and
    2) all the bytes in the IKEv2-SCSI payload-specific data part (see 5.3.5.1) of the Identification – Device Server payload (see 5.3.5.4).

After the Authentication step SECURITY PROTOCOL IN command completes with GOOD status, the application client should verify the contents of the AUTHENTICATION DATA field by applying the algorithm specified by the ALGORITHM IDENTIFIER field in the SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) in the Key Exchange step (see 4.1.3.6) as described in table 72 (see 5.3.6.6) and this subclause to generate the following concatenation of bytes:

1) all the bytes in the Data-In Buffer returned by the Device Server Capabilities step (see 4.1.3.5) SECURITY PROTOCOL IN command;
2) all the bytes in the Data-In Buffer returned by the most recent Key Exchange step SECURITY PROTOCOL IN command (see 4.1.3.6.3) in the same IKEv2-SCSI CCS for which GOOD status was returned;
3) all the bytes in the IKEv2-SCSI payload-specific data part (see 5.3.5.1) of the Nonce payload (see 5.3.5.8) sent in the Key Exchange step SECURITY PROTOCOL OUT command in the same IKEv2-SCSI CCS; and
4) all the bytes produced by applying the PRF selected by the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) to the following inputs:
    1) the SK_pr shared key (see 4.1.3.4); and
    2) all the bytes in the IKEv2-SCSI payload-specific data part (see 5.3.5.1) of the Identification – Device Server payload (see 5.3.5.4) received in the SECURITY PROTOCOL IN parameter data.

If the verification of the contents of the AUTHENTICATION DATA field is not successful, then the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 4.1.3.10.

If the AUTH METHOD field is set to 01h (i.e., RSA Digital Signature), the RSA digital signature shall be encoded with the EMSA-PKCS1-v1_5 signature encoding method as defined in RFC 3447 (see RFC 7296).

5.3.5.8 Nonce payload

The Nonce payload (see table 52) transfers one random nonce from the application client to the device server or from the device server to the application client.

Table 52 — Nonce payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	NONCE DATA							
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 5.3.5.1.

In the Key Exchange step SECURITY PROTOCOL OUT command (see 4.1.3.6.2) the NONCE DATA field specifies the application client's random nonce.

In the Key Exchange step SECURITY PROTOCOL IN command (see 4.1.3.6.3) the NONCE DATA field indicates the device server's random nonce.

The requirements that RFC 7296 places on the nonce data shall apply to this standard.

5.3.5.9 Notify payload

This standard uses the Notify payload (see table 53) to provide initial contact notification from the application client to the device server. See Annex B for information about differences between this standard and IKEv2.

Table 53 — Notify payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (0010h)						
3								(LSB)
4	PROTOCOL ID (01h)							
5	SAI SIZE (08h)							
6	(MSB)	NOTIFY MESSAGE TYPE (4000h)						
7								(LSB)
8		Restricted (see RFC 7296)						
•••								
11								
12	(MSB)	SAI						
•••								
15								(LSB)

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 5.3.5.1.

The PROTOCOL ID field is set as shown in table 53 for the IKEv2-SCSI Notify payload. If the device server receives a PROTOCOL ID field that is not set as shown in table 53, the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 5.3.8.3.

The SAI SIZE field is set as shown in table 53 for the IKEv2-SCSI Notify payload. If the device server receives an SAI SIZE field that is not set as shown in table 53, the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 5.3.8.3.

The NOTIFY MESSAGE TYPE field is set as shown in table 53 (i.e., INITIAL_CONTACT) for the IKEv2-SCSI Notify payload. If the device server receives a NOTIFY MESSAGE TYPE field that is not set as shown in table 53, the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 5.3.8.3.

The SAI field specifies the device server's SAI. If the contents of the SAI field are not identical to the contents of the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header (see 5.3.4), then the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 5.3.8.3.

Unless an error is detected, the device server shall process the Notify payload as described in 4.1.3.7.2.

5.3.5.10 Delete payload

The Delete payload (see table 54) requests the deletion of an existing SA or the abandonment of an IKEv2-SCSI CCS that is in progress.

Table 54 — Delete payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (0018h)						
3								(LSB)
4	PROTOCOL ID (01h)							
5	SAI SIZE (08h)							
6	(MSB)	NUMBER OF SAIS (0002h)						
7								(LSB)
8		Restricted (see RFC 7296)						
•••								
11								
12	(MSB)	AC_SAI						
•••								
15								(LSB)
16		Restricted (see RFC 7296)						
•••								
19								
20	(MSB)	DS_SAI						
•••								
23								(LSB)

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 5.3.5.1.

The PROTOCOL ID field is set as shown in table 54 for the IKEv2-SCSI Delete payload. If the device server receives a PROTOCOL ID field that is not set as shown in table 54, the error shall be processed as described in 5.3.8.3.

The SAI SIZE field is set as shown in table 54 for the IKEv2-SCSI Delete payload. If the device server receives an SAI SIZE field that is not set as shown in table 54, the error shall be processed as described in 5.3.8.3.

The NUMBER OF SAIS field is set as shown in table 54 for the IKEv2-SCSI Delete payload. If the device server receives a NUMBER OF SAIS field that is not set as shown in table 54, the error shall be processed as described in 5.3.8.3.

The AC_SAI field specifies the AC_SAI SA parameter value for the SA to be deleted. If the contents of the AC_SAI field do not match the contents of the IKE_SA APPLICATION CLIENT SAI field in the IKEv2-SCSI header (see 5.3.4), then the error shall be processed as described in 5.3.8.3.

The DS_SAI field specifies the DS_SAI SA parameter value for the SA to be deleted. If the contents of the DS_SAI field do not match the contents of the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header (see 5.3.4), then the error shall be processed as described in 5.3.8.3.

If the device server is maintaining SA parameters for which the AC_SAI matches the contents of the AC_SAI field and the DS_SAI matches the contents of the DS_SAI field, then that set of SA parameters shall be deleted.

The IKEv2-SCSI CCS shall be abandoned (see 4.1.3.10) if:

a) the device server is maintaining state for an IKEv2-SCSI CCS on the I_T_L nexus on which the command was received; and
b) the contents of:
   A) the AC_SAI field match the application client's SAI in the maintained state; and
   B) the DS_SAI field match the device server's SAI in the maintained state.

5.3.5.11 Encrypted payload

5.3.5.11.1 Combined mode encryption

The following types of encryption algorithms are used in the Encrypted payload:

a) non-combined modes that use separate algorithms to encrypt/decrypt and integrity check the Encrypted payload; and
b) combined modes in which a single encryption algorithm does both encryption/decryption and integrity checking.

The ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.4) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) in the Key Exchange step (see 4.1.3.6) indicates the type of encryption algorithm to be applied to the Encrypted payload for IKEv2-SCSI functions as follows:

a) if the ALGORITHM IDENTIFIER field is not set to AUTH_COMBINED, a non-combined mode encryption algorithm is being used; or
b) if the ALGORITHM IDENTIFIER field is set to AUTH_COMBINED, a combined mode encryption algorithm is being used.

If the Encrypted payload is in parameter data that is not associated with an active IKEv2-SCSI CCS, then the integrity checking algorithm identifier that selects between combined and non-combined mode encryption is found in the MGMT_DATA SA parameter (see 4.1.3.9).

5.3.5.11.2 Encrypted payload introduction

The Encrypted payload (see table 55) transfers one or more other IKEv2-SCSI payloads that are encrypted and integrity checked from the application client to the device server and vice versa.

If IKEv2-SCSI parameter data contains the Encrypted payload, then the Encrypted payload is the first payload in the parameter data (i.e., the NEXT PAYLOAD field in the IKEv2-SCSI header (see 5.3.5.1) contains 2Eh). Since the NEXT PAYLOAD field in an Encrypted payload identifies the first payload in the CIPHERTEXT field, there is no way to identify a payload following the Encrypted payload. As a result, the Encrypted payload is required to be the last payload in the IKEv2-SCSI payloads part of the parameter data (see 5.3.4).

Table 55 — Encrypted payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	INITIALIZATION VECTOR							
•••								
i-1								
i	CIPHERTEXT							
•••								
k-1								
k	INTEGRITY CHECK VALUE							
•••								
n								

The NEXT PAYLOAD field identifies the first IKEv2-SCSI payload in the CIPHERTEXT field using the coded values shown in table 41 (see 5.3.5.1).

The CRIT bit and IKE PAYLOAD LENGTH field are described in 5.3.5.1.

The IKE PAYLOAD LENGTH field specifies the number of bytes that follow in the Encrypted payload. The number of bytes in the CIPHERTEXT field is equal to the number of bytes in the plaintext (see table 56).

The INITIALIZATION VECTOR field specifies the initialization vector encryption algorithm input value. The size of the initialization vector is defined by the encryption algorithm as shown in table 64 (see 5.3.6.2).

The CIPHERTEXT field contains an output of the encryption algorithm specified by the ALGORITHM IDENTIFIER field in the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.2) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) in the Key Exchange step (see 4.1.3.6) using the inputs:

a) IKEv2-SCSI AAD is an encryption algorithm input as follows:
   A) if a non-combined mode encryption algorithm is being used (see 5.3.5.11.1), then there is no AAD input; or
   B) if a combined mode encryption algorithm is being used (see 5.3.5.11.1), then the AAD described in 5.3.5.11.3 is an input;
b) the contents of the INITIALIZATION VECTOR field (see table 64);

c) the plaintext data shown in table 56; and
d) the shared key value, key length, and salt bytes (see table 64 in 5.3.6.2), if any, for one of the following shared keys:
   A) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL OUT command (see 4.1.3.7.2) or the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 4.1.3.11) that affects an active IKEv2-SCSI CCS, then the SK_ei shared key (see 4.1.3.4) for the IKEv2-SCSI CCS;
   B) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL IN command (see 4.1.3.7.3), then the SK_er shared key (see 4.1.3.4) for the IKEv2-SCSI CCS; or
   C) if the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 4.1.3.11) that does not affect an active IKEv2-SCSI CCS, then the SK_ei shared key (see 4.1.3.4) from the MGMT_DATA SA parameter (see 4.1.3.9).

NOTE 10 - Salt bytes (see table 64 in 5.3.6.2) are used only by combined mode encryption algorithms (see 5.3.5.11.1).

If the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command that does not affect an active IKEv2-SCSI CCS, then the encryption algorithm identifier stored in the MGMT_DATA SA parameter indicates the encryption algorithm to use.

The INTEGRITY CHECK VALUE field contains the integrity check value that is computed as described in 5.3.5.11.1 (e.g., if the integrity algorithm is AUTH_COMBINED, then the integrity check value is an output of the encryption algorithm). The size of the integrity check value is defined by the integrity algorithm or encryption algorithm, depending on which algorithm computes the value.

If the integrity algorithm specified by the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) in the Key Exchange step is not AUTH_COMBINED, then the contents of the INTEGRITY CHECK VALUE field are computed by processing the integrity check algorithm specified by the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor using the following inputs:

a) a byte string composed of:
   1) the AAD described in 5.3.5.11.3;
   2) the contents of the INITIALIZATION VECTOR field (see table 55); and
   3) the ciphertext that is the result of encrypting the plaintext data;
   and
b) the shared key value from one of the following shared keys:
   A) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL OUT command (see 4.1.3.7.2) or the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 4.1.3.11) that affects an active IKEv2-SCSI CCS, then the SK_ai shared key (see 4.1.3.4) for the IKEv2-SCSI CCS;
   B) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL IN command (see 4.1.3.7.3), then the SK_ar shared key (see 4.1.3.4) for the IKEv2-SCSI CCS; or
   C) if the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 4.1.3.11) that does not affect an active IKEv2-SCSI CCS, then the SK_ai shared key (see 4.1.3.4) from the MGMT_DATA SA parameter (see 4.1.3.9).

If the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command that does not affect an active IKEv2-SCSI CCS, then the integrity checking algorithm identifier stored in the MGMT_DATA SA parameter indicates the integrity checking algorithm to use.

While computing the encrypted CIPHERTEXT field contents for an Encrypted payload, the plaintext format shown in table 56 is used.

Table 56 — Plaintext format for Encrypted payload CIPHERTEXT field

Bit / Byte	7	6	5	4	3	2	1	0
IKEv2-SCSI payloads								
0	IKEv2-SCSI payload (see 5.3.5) [first]							
•••								
	⋮							
	IKEv2-SCSI payload (see 5.3.5) [last]							
•••								
n								
p	PADDING BYTES (if any)							
•••								
k-1								
k	PAD LENGTH (k-p)							

Each IKEv2-SCSI payload (see 5.3.5) contains specific data related to the operation being performed. A specific combination of IKEv2-SCSI payloads is required for each operation (e.g., Authentication) as summarized in 4.1.3.2.

The PADDING BYTES field contains zero to 255 bytes. The number of padding bytes is:

a) defined by the encryption algorithm; or
b) any number of bytes that causes the length of all plaintext bytes (i.e., k+1) to be a whole multiple of the alignment (see table 64 in 5.3.6.2) for the encryption algorithm being used.

The contents of the padding bytes are:

a) defined by the encryption algorithm; or
b) if the encryption algorithm does not define the padding bytes contents, a series of one byte binary values starting at one and incrementing by one in each successive byte (i.e., 01h in the first padding byte, 02h in the second padding byte, etc.).

If the encryption algorithm does not place requirements on the contents of the padding bytes (i.e., option b) is in effect), then after decryption the contents of the padding bytes shall be verified to match the series of one byte binary values described in this subclause. If this verification is not successful in a device server, the error shall be processed as described in 5.3.8.2. If this verification is not successful in an application client, the decrypted data should be ignored.

The PAD LENGTH field specifies the number of bytes in the PADDING BYTES field.

5.3.5.11.3 IKEv2-SCSI AAD

The AAD defined by this standard for IKEv2-SCSI use is as follows:

1) all the bytes in the IKEv2-SCSI Header (see 5.3.4); and
2) all the bytes in the IKEv2-SCSI Payload Header (see 5.3.5.1) of the Encrypted payload (see 5.3.5.11).

5.3.5.11.4 Processing a received Encrypted payload

Before performing any checks of data contained in the CIPHERTEXT field, the contents of the INTEGRITY CHECK VALUE field and CIPHERTEXT field shall be integrity checked and decrypted based on the contents of the IKE_SA APPLICATION CLIENT SAI field and the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header (see 5.3.4) as described in this subclause.

Computation of the comparison integrity check value and decryption of an Encrypted payload is performed as follows:

1) if a non-combined mode encryption algorithm is being used (see 5.3.5.11.1), then the comparison integrity check value is computed by performing the integrity check algorithm specified by the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor using the following inputs:
   A) a byte string composed of:
      1) the AAD described in 5.3.5.11.3;
      2) the contents of the INITIALIZATION VECTOR field (see table 55) in the Encrypted payload; and
      3) the contents of the CIPHERTEXT field (see table 55) in the Encrypted payload;

      and
   B) the shared key value from one of the following shared keys:
      a) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL OUT command (see 4.1.3.7.2) or the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 4.1.3.11) that affects an active IKEv2-SCSI CCS, then the SK_ai shared key (see 4.1.3.4) for the IKEv2-SCSI CCS;
      b) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL IN command (see 4.1.3.7.3), then the SK_ar shared key (see 4.1.3.4) for the IKEv2-SCSI CCS; or
      c) if the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 4.1.3.11) that does not affect an active IKEv2-SCSI CCS, then the SK_ai shared key (see 4.1.3.4) from the MGMT_DATA SA parameter (see 4.1.3.9);

   and
2) process the CIPHERTEXT field using the encryption algorithm specified by the ALGORITHM IDENTIFIER field in the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.2) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) in the Key Exchange step (see 4.1.3.6) using the inputs:
   A) IKEv2-SCSI AAD is an input to the encryption algorithm as follows:
      a) if a non-combined mode encryption algorithm is being used (see 5.3.5.11.1), then there is no AAD input; or
      b) if a combined mode encryption algorithm is being used (see 5.3.5.11.1), then the AAD described in 5.3.5.11.3 is an input;
   B) the contents of the INITIALIZATION VECTOR field (see table 55) in the Encrypted payload;
   C) the contents of the CIPHERTEXT field (see table 55) in the Encrypted payload; and
   D) the shared key value, key length, and salt bytes (see table 64 in 5.3.6.2), if any, for one of the following shared keys:
      a) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL OUT command (see 4.1.3.7.2), or the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 4.1.3.11) that affects an active IKEv2-SCSI

CCS, then the SK_ei shared key (see 4.1.3.4) for the IKEv2-SCSI CCS;
b) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL IN command (see 4.1.3.7.3), then the SK_er shared key (see 4.1.3.4) for the IKEv2-SCSI CCS; or
c) if the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 4.1.3.11) that does not affect an active IKEv2-SCSI CCS, then the SK_ei shared key (see 4.1.3.4) from the MGMT_DATA SA parameter (see 4.1.3.9).

The results of step 2) are:

a) the decrypted plaintext, if a non-combined mode encryption algorithm is being used (see 5.3.5.11.1); or
b) the decrypted plaintext and the comparison integrity check value, if a combined mode encryption algorithm is being used.

If the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command that does not affect an active IKEv2-SCSI CCS, then the integrity checking algorithm identifier value and encryption algorithm identifier value that are stored in the MGMT_DATA SA parameter indicate the integrity checking algorithm to use.

If the comparison integrity check value differs from the value in the INTEGRITY CHECK VALUE field of the Encrypted payload, then:

a) the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 4.1.3.10; and
b) the device server shall respond to the mismatch as follows:
   A) if the IKEv2-SCSI header (see 5.3.4) specifies an attempt to provide authentication data for or the deletion of an IKE-v2-SCSI CCS on the I_T_L nexus on which the command was received, then the device server shall:
      a) terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE REJECTED; and
      b) continue the IKEv2-SCSI CCS by preparing to receive another Authentication step SECURITY PROTOCOL OUT command;

      or
   B) if the IKEv2-SCSI header (see 5.3.4) specifies the deletion of an active SA, then the device server shall terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

5.3.5.12 IKEv2-SCSI SA Creation Capabilities payload

The IKEv2-SCSI SA Creation Capabilities payload (see table 57) lists all the security algorithms that the device server allows to be used in an IKEv2-SCSI CCS. Events that are outside the scope of this standard may change the contents of the IKEv2-SCSI SA Creation Capabilities payload at any time.

Table 57 — IKEv2-SCSI SA Creation Capabilities payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD (00h)							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4		Reserved						
5								
6								
7	NUMBER OF ALGORITHM DESCRIPTORS							
IKEv2-SCSI cryptographic algorithm descriptors								
8	IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6) [first]							
•••								
	⋮							
	IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6) [last]							
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 5.3.5.1.

The NEXT PAYLOAD field is set as shown in table 57 (i.e., No Next Payload) in the IKEv2-SCSI SA Creation Capabilities payload.

The NUMBER OF ALGORITHM DESCRIPTORS field specifies the number of IKEv2-SCSI cryptographic algorithm descriptors that follow in the IKEv2-SCSI SA Creation Capabilities payload.

Each IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6) describes one combination of security algorithm and algorithm attributes that the device server allows to be used in an IKEv2-SCSI CCS. If more than one set of algorithm attributes (e.g., key length) is allowed for any allowed security algorithm, then a different SCSI cryptographic algorithms descriptor shall be included for each set of algorithm attributes.

The SCSI cryptographic algorithms descriptors shall be ordered by:

1) increasing algorithm type;
2) increasing algorithm identifier within the same algorithm type; and
3) increasing key length, if any, within the same algorithm identifier.

The algorithms allowed may be a subset of the algorithms supported by the device server.

The method for changing the device server supported algorithms that are allowed is outside the scope of this standard. Any changes in allowed algorithms do not take effect until the new list is returned to any application client in an IKEv2-SCSI SA Creation Capabilities payload.

5.3.5.13 IKEv2-SCSI SA Cryptographic Algorithms payload

The IKEv2-SCSI SA Cryptographic Algorithms payload (see table 58) lists the security algorithms that are being used in the creation and management (e.g., deletion) of an SA using an IKEv2-SCSI CCS.

Table 58 — IKEv2-SCSI SA Cryptographic Algorithms payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	Reserved							
•••								
13								
14	(MSB)	USAGE DATA LENGTH (0000h)						
15								(LSB)
16	Reserved							
•••								
19								
20	NUMBER OF ALGORITHM DESCRIPTORS							
IKEv2-SCSI cryptographic algorithm descriptors								
21	IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6) [first]							
•••								
	⋮							
	IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6) [last]							
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 5.3.5.1.

The USAGE DATA LENGTH field is set as shown in table 58 in the IKEv2-SCSI SA Cryptographic Algorithms payload.

The NUMBER OF ALGORITHM DESCRIPTORS field specifies the number of IKEv2-SCSI cryptographic algorithm descriptors that follow in the IKEv2-SCSI SA Cryptographic Algorithms payload. If a device server receives a NUMBER OF ALGORITHM DESCRIPTORS field that is not set to six, then the error shall be processed as described in 5.3.8.3.

Each IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6) describes one combination of security algorithm and algorithm attributes to be used during the IKEv2-SCSI CCS.

The IKEv2-SCSI cryptographic algorithm descriptors are ordered as follows:

1) one ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.2);
2) one PRF IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.3);
3) one INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.4);
4) one D-H IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.5);
5) one SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.6); and
6) one SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.6).

If a device server receives an IKEv2-SCSI SA Cryptographic Algorithms payload that does not contain the IKEv2-SCSI cryptographic algorithm descriptors described in this subclause in the order described in this subclause, then the error shall be processed as described in 5.3.8.3.

If the device server receives an IKEv2-SCSI SA Cryptographic Algorithms payload that contains an ENCR IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to ENCR_NULL, then the error shall be processed as described in 5.3.8.3.

In the Key Exchange step SECURITY PROTOCOL IN parameter data (see 4.1.3.6.3), the device server returns the IKEv2-SCSI SA Cryptographic Algorithms payload received during the Key Exchange step SECURITY PROTOCOL OUT command (see 4.1.3.6.2) to confirm acceptance of the algorithms.

5.3.5.14 IKEv2-SCSI SAUT Cryptographic Algorithms payload

The IKEv2-SCSI SAUT Cryptographic Algorithms payload (see table 59) lists the usage type of and security algorithms to be used by the SA that is created as a result of an IKEv2-SCSI CCS.

Table 59 — IKEv2-SCSI SAUT Cryptographic Algorithms payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4		Reserved						
11								
12	(MSB)	SA TYPE						
13								(LSB)
14	(MSB)	USAGE DATA LENGTH (j)						
15								(LSB)
16		USAGE DATA						
•••								
16+j-1								
16+j		Reserved						
16+j+1								
16+j+2								
16+j+3	NUMBER OF ALGORITHM DESCRIPTORS							
IKEv2-SCSI cryptographic algorithm descriptors								
16+j+4		IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6) [first]						
•••								
		⋮						
		IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6) [last]						
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 5.3.5.1.

The SA TYPE field specifies the usage type for the SA and is selected from among those listed in table 4 (see 4.1.1.2). An error shall be processed as described in 5.3.8.3 if any of the following conditions occur:

a) the device server receives an SA usage type that is not allowed by the device server;
b) an SA usage type between 8000h and BFFFh inclusive is received in a Key Exchange step SECURITY PROTOCOL OUT command (see 4.1.3.6.2); or
c) an SA usage type between A000h and CFFFh inclusive is received for which the device server is unable to verify the applicable usage type constraints.

The method for changing the device server supported SA usage types that are allowed is outside the scope of this standard.

The USAGE DATA LENGTH field specifies number of bytes of usage data that follow.

The size and format of the usage data depends on the SA type (see table 4 in 4.1.1.2). If the device server receives a USAGE DATA LENGTH field that contains a value that is inconsistent with the SA type, then the error shall be processed as described in 5.3.8.3.

The USAGE DATA field contains information to be stored in the USAGE_DATA SA parameter if the SA is generated (see 4.1.3.9).

The NUMBER OF ALGORITHM DESCRIPTORS field specifies the number of IKEv2-SCSI cryptographic algorithm descriptors that follow in the IKEv2-SCSI SAUT Cryptographic Algorithms payload. If a device server receives a NUMBER OF ALGORITHM DESCRIPTORS field that is not set to two, the error shall be processed as described in 5.3.8.3.

Each IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6) describes one combination of security algorithm and algorithm attributes to be used by the SA created as a result of the IKEv2-SCSI CCS. The IKEv2-SCSI cryptographic algorithm descriptors are ordered as follows:

1) one ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.2); and
2) one INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.4).

If a device server receives an IKEv2-SCSI SAUT Cryptographic Algorithms payload that does not contain the IKEv2-SCSI cryptographic algorithm descriptors described in this subclause in the order described in this subclause, then the error shall be processed as described in 5.3.8.3.

5.3.5.15 IKEv2-SCSI Timeout Values payload

The IKEv2-SCSI Timeout Values payload (see table 60) specifies the timeout intervals associated with an IKEv2-SCSI CCS.

Table 60 — IKEv2-SCSI Timeout Values payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (0010h)						
3								(LSB)
4	Reserved							
•••								
6								
7	NUMBER OF TIMEOUT VALUES (02h)							
8	(MSB)	IKEV2-SCSI PROTOCOL TIMEOUT						
•••								
11								(LSB)
12	(MSB)	IKEV2-SCSI SA INACTIVITY TIMEOUT						
•••								
15								(LSB)

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 5.3.5.1.

The NUMBER OF TIMEOUT VALUES field specifies the number of four-byte timeout values that follow. If the number of timeout values is less than two, the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 5.3.8.3.

The IKEV2-SCSI PROTOCOL TIMEOUT field specifies the number of seconds that the device server shall wait for the next command in the IKEv2-SCSI CCS. If the timeout expires before the device server receives an IKEv2-SCSI CCS command, the device server shall abandon the IKEv2-SCSI CCS as described in 4.1.3.10.

The IKEV2-SCSI SA INACTIVITY TIMEOUT field specifies the number of seconds that the device server shall wait for the next command that uses an SA. This value is copied to the TIMEOUT SA parameter when the SA is generated (see 4.1.3.9).

The device server shall replace any timeout value that is set to zero with a value of ten (i.e., ten seconds).

The maximum value for the protocol timeout should be long enough to allow the application client to continue the IKEv2-SCSI CCS, but short enough that, if an incomplete IKEv2-SCSI CCS is abandoned, the device server discards the state for that IKEv2-SCSI CCS and becomes available to for another IKEv2-SCSI CCS without delays that have adverse effects.

5.3.6 IKEv2-SCSI cryptographic algorithm descriptors

5.3.6.1 Overview

Each IKEv2-SCSI cryptographic algorithm descriptor (see table 61) specifies a cryptographic algorithm for IKEv2-SCSI (e.g., an encryption algorithm, an integrity checking algorithm, a key generation algorithm, or an authentication algorithm).

Table 61 — IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE							
1	Reserved							
2	(MSB)			IKE DESCRIPTOR LENGTH (000Ch)				
3								(LSB)
4	(MSB)			ALGORITHM IDENTIFIER				
•••								
7								(LSB)
8				ALGORITHM ATTRIBUTES				
•••								
11								

The ALGORITHM TYPE field (see table 62) specifies the type of cryptographic algorithm that applies to this IKEv2-SCSI cryptographic algorithm descriptor.

Table 62 — ALGORITHM TYPE field

Code	Name	Description	Reference
01h	ENCR	Encryption algorithm	5.3.6.2
02h	PRF	Pseudo-random function	5.3.6.3
03h	INTEG	Integrity algorithm	5.3.6.4
04h	D-H	Diffie-Hellman group	5.3.6.5
05h to F0h	Restricted		RFC 7296
F9h	SA_AUTH_OUT	IKEv2-SCSI authentication algorithm for SECURITY PROTOCOL OUT data	5.3.6.6
FAh	SA_AUTH_IN	IKEv2-SCSI authentication algorithm for SECURITY PROTOCOL IN data	5.3.6.6
All others	Reserved		

The IKE DESCRIPTOR LENGTH field specifies the total number of bytes in the IKEv2-SCSI SA Cryptographic Algorithms descriptor and shall be set as shown in table 61 in all the IKEv2-SCSI SA Cryptographic Algorithms descriptors summarized in table 62.

NOTE 11 - The contents of the IKE DESCRIPTOR LENGTH field differ from those found in most SCSI length fields, However, they are consistent with the IKEv2 usage (see RFC 7296).

The contents of the ALGORITHM IDENTIFIER field and ALGORITHM ATTRIBUTES field depend on the contents of the ALGORITHM TYPE field (see table 62). The ALGORITHM ATTRIBUTES field is reserved in some IKEv2-SCSI SA Cryptographic Algorithms descriptor formats.

5.3.6.2 ENCR IKEv2-SCSI cryptographic algorithm descriptor

If the ALGORITHM TYPE field is set to ENCR (i.e., 01h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 63), then the descriptor specifies an encryption algorithm to be applied during the IKEv2-SCSI Authentication step (see 4.1.3.7), SA deletion operation (see 4.1.3.11), and when the SA created by the IKEv2-SCSI is applied to user data.

Table 63 — ENCR IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (01h)							
1	Reserved							
2	(MSB)			IKE DESCRIPTOR LENGTH (000Ch)				
3								(LSB)
4	(MSB)			ALGORITHM IDENTIFIER				
•••								
7								(LSB)
8				Reserved				
9								
10	(MSB)			KEY LENGTH				
11								(LSB)

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 5.3.6.1, and shall be set as shown in table 63 for the ENCR IKEv2-SCSI cryptographic algorithm descriptor.

The ALGORITHM IDENTIFIER field (see table 64) specifies the encryption algorithm that applies to this ENCR IKEv2-SCSI cryptographic algorithm descriptor.

Table 64 — ENCR ALGORITHM IDENTIFIER field

Code	Description	Salt bytes length (bytes)	IV [a] length (bytes)	Align-ment [b] (bytes)	Key length (bytes)	Support	Reference
8001 000Bh	ENCR_NULL	n/a	0	4	0	Mandatory	RFC 2410
8001 000Ch	AES-CBC	n/a	16	16	16	Optional	RFC 3602
					24	Prohibited	
					32	Optional	
8001 0010h	AES-CCM with a 16-byte MAC	3	8	4	16	Optional	RFC 4309 and RFC 7296
					24	Prohibited	
					32	Optional	
8001 0014h	AES-GCM with a 16-byte MAC	4	8	4	16	Optional	RFC 4106 and RFC 7296
					24	Prohibited	
					32	Optional	
8001 0400h to 8001 FFFFh	Vendor Specific						
0000 0000h to 0000 FFFFh	Restricted						IANA
All others	Reserved						

[a] Initialization Vector.
[b] The alignment required in the plaintext prior to encryption.

ENCR_NULL indicates that encryption shall not be applied when the SA created by the IKEv2-SCSI is applied to user data.

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned and an error shall be processed as described in 5.3.8.3 if the parameter list contains an IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) that contains:

a) an ENCR IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to ENCR_NULL;
b) the following combination of IKEv2-SCSI cryptographic algorithm descriptors (see 5.3.6.4):
   A) an INTEG IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value other than AUTH_COMBINED; and
   B) an ENCR IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value that table 64 describes as requiring AUTH_COMBINED as the integrity check algorithm;

   or
c) the following combination of IKEv2-SCSI cryptographic algorithm descriptors:
   A) an INTEG IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to AUTH_COMBINED; and

B) an ENCR IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value that table 64 does not describe as requiring AUTH_COMBINED as the integrity check algorithm.

The KEY LENGTH field specifies the number of bytes in the shared key for the encryption algorithm that applies to this ENCR IKEv2-SCSI cryptographic algorithm descriptor.

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 5.3.8.3 if the parameter list contains:

a) no ENCR IKEv2-SCSI cryptographic algorithm descriptors;
b) more than one ENCR IKEv2-SCSI cryptographic algorithm descriptor;
c) an ENCR IKEv2-SCSI cryptographic algorithm descriptor that does not appear in the SA Creations Capabilities payload (see 5.3.5.12) last returned by the device server to any application client (i.e., an ENCR IKEv2-SCSI cryptographic algorithm descriptor for a combination of algorithm identifier and key length that the device server has not reported as one of its SA creation capabilities); or
d) an ENCR IKEv2-SCSI cryptographic algorithm descriptor that contains:
   A) an algorithm identifier that is not shown in table 64; or
   B) a key length that:
      a) does not match one of the values shown in table 64; or
      b) is not supported by the device server.

5.3.6.3 PRF IKEv2-SCSI cryptographic algorithm descriptor

If the ALGORITHM TYPE field is set to PRF (i.e., 02h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 65), then the descriptor specifies the pseudo-random function and KDF to be used during the Key Exchange step completion (see 4.1.3.6.4).

Table 65 — PRF IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (02h)							
1	Reserved							
2	(MSB)							
3	IKE DESCRIPTOR LENGTH (000Ch)							(LSB)
4	(MSB)							
•••	ALGORITHM IDENTIFIER							
7								(LSB)
8								
•••	Reserved							
11								

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 5.3.6.1, and shall be set as shown in table 65 for the PRF IKEv2-SCSI cryptographic algorithm descriptor.

The ALGORITHM IDENTIFIER field (see table 66) specifies PRF and KDF that applies to this PRF IKEv2-SCSI cryptographic algorithm descriptor.

Table 66 — PRF ALGORITHM IDENTIFIER field

Code	Description	Support	Output length (bytes)	Reference	
				PRF [a]	KDF [b]
8002 0002h	IKEv2-use based on SHA-1	Optional	20	RFC 6151	4.1.2.3
8002 0004h	IKEv2-use based on AES-128 in CBC mode	Optional	16	RFC 4434	4.1.2.4
8002 0005h	IKEv2-use based on SHA-256	Optional	32	RFC 4868	4.1.2.3
8002 0007h	IKEv2-use based on SHA-512	Optional	64	RFC 4868	4.1.2.3
8002 0400h to 8002 FFFFh	Vendor Specific				
0000 0000h to 0000 FFFFh	Restricted			IANA	
All others	Reserved				

[a] PRFs are equivalent to the prf() functions defined in RFC 7296.
[b] KDFs are equivalent to the prf+() functions defined in RFC 7296.

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 5.3.8.3 if the parameter list contains:

a) no PRF IKEv2-SCSI cryptographic algorithm descriptors;
b) more than one PRF IKEv2-SCSI cryptographic algorithm descriptor;
c) a PRF IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload (see 5.3.5.12) last returned by the device server to any application client; or
d) an PRF IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 66.

5.3.6.4 INTEG IKEv2-SCSI cryptographic algorithm descriptor

If the ALGORITHM TYPE field is set to INTEG (i.e., 03h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 67), then the descriptor specifies an integrity checking (i.e., data authentication) algorithm to be applied during the IKEv2-SCSI Authentication step (see 4.1.3.7) and when the SA created by the IKEv2-SCSI is applied to user data.

Table 67 — INTEG IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (03h)							
1	Reserved							
2	(MSB)			IKE DESCRIPTOR LENGTH (000Ch)				
3								(LSB)
4	(MSB)			ALGORITHM IDENTIFIER				
•••								
7								(LSB)
8				Reserved				
•••								
11								

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 5.3.6.1, and shall be set as shown in table 67 for the INTEG IKEv2-SCSI cryptographic algorithm descriptor.

The ALGORITHM IDENTIFIER field (see table 68) specifies integrity checking algorithm and shared key length that applies to this INTEG IKEv2-SCSI cryptographic algorithm descriptor.

Table 68 — INTEG ALGORITHM IDENTIFIER field

Code	IKEv2 Name	ICV [a] length (bytes)	Key length (bytes)	Support	Reference
8003 0002h	AUTH_HMAC_SHA1_96	12	20	Optional	RFC 2404
8003 000Ch	AUTH_HMAC_SHA2_256_128	16	32	Optional	RFC 4868
8003 000Eh	AUTH_HMAC_SHA2_512_256	32	64	Optional	RFC 4868
F003 0001h	AUTH_COMBINED	n/a	0	Optional	this subclause
8003 0400h to 8003 FFFFh	Vendor Specific				
0000 0000h to 0000 FFFFh	Restricted				IANA
All others	Reserved				

[a] Integrity Check Value.

The AUTH_COMBINED integrity checking algorithm is used with encryption algorithms that include integrity checking as described in 5.3.6.2. The AUTH_COMBINED algorithm identifier specifies that no additional integrity check is performed, as indicated by the zero-length key.

The key length used with an integrity checking algorithm is determined by the algorithm identifier as shown in table 68.

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 5.3.8.3 if the parameter list contains:

a) no INTEG IKEv2-SCSI cryptographic algorithm descriptors;
b) more than one INTEG IKEv2-SCSI cryptographic algorithm descriptor;
c) an INTEG IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload (see 5.3.5.12) last returned by the device server to any application client; or
d) an INTEG IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 68.

5.3.6.5 D-H IKEv2-SCSI cryptographic algorithm descriptor

If the ALGORITHM TYPE field is set to D-H (i.e., 04h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 69), then the descriptor specifies Diffie-Hellman group and Diffie-Hellman algorithm used during the IKEv2-SCSI Key Exchange step (see 4.1.3.6) to derive a shared key that is known only to the application client and device server.

Table 69 — D-H IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (04h)							
1	Reserved							
2	(MSB)			IKE DESCRIPTOR LENGTH (000Ch)				
3								(LSB)
4	(MSB)			ALGORITHM IDENTIFIER				
•••								
7								(LSB)
8				Reserved				
•••								
11								

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 5.3.6.1, and shall be set as shown in table 69 for the D-H IKEv2-SCSI cryptographic algorithm descriptor.

The ALGORITHM IDENTIFIER field (see table 70) specifies Diffie-Hellman algorithm, group, and shared key length that applies to this D-H IKEv2-SCSI cryptographic algorithm descriptor.

Table 70 — D-H ALGORITHM IDENTIFIER field

Code	Description	Key length (bytes)	Support	Reference
8004 000Eh	2 048-bit MODP group (finite field D-H)	256	Optional	RFC 3526
8004 000Fh	3 072-bit MODP group (finite field D-H)	384	Optional	RFC 3526
8004 0010h	4 096-bit MODP group (finite field D-H)	512	Optional	RFC 3526
8004 0011h	6 144-bit MODP group (finite field D-H)	768	Optional	RFC 3526
8004 0012h	8 192-bit MODP group (finite field D-H)	1 024	Optional	RFC 3526
8004 0013h	256-bit random ECP group	64	Optional	RFC 5903
8004 0015h	521-bit random ECP group	132	Optional	RFC 5903
8004 0400h to 8004 FFFFh	Vendor Specific			
0000 0000h to 0000 FFFFh	Restricted			IANA
All others	Reserved			

The key length of the public value transferred in the KEY EXCHANGE DATA field (see 5.3.5.3) is determined by the algorithm identifier as shown in table 70.

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 5.3.8.3 if the parameter list contains:

a) no D-H IKEv2-SCSI cryptographic algorithm descriptors;
b) more than one D-H IKEv2-SCSI cryptographic algorithm descriptor;
c) a D-H IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload (see 5.3.5.12) last returned by the device server to any application client; or
d) an D-H IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 70.

5.3.6.6 IKEv2-SCSI authentication algorithm IKEv2-SCSI cryptographic algorithm descriptor

If the ALGORITHM TYPE field is set to SA_AUTH_OUT (i.e., F9h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 71), then the descriptor specifies Authentication payload authentication algorithm used by the IKEv2-SCSI Authentication step SECURITY PROTOCOL OUT command (see 4.1.3.7.2).

If the ALGORITHM TYPE field is set to SA_AUTH_IN (i.e., FAh) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 71), then the descriptor specifies Authentication payload authentication algorithm used by the IKEv2-SCSI Authentication step SECURITY PROTOCOL IN command (see 4.1.3.7.3).

Table 71 — SA_AUTH_OUT and SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (F9h or FAh)							
1	Reserved							
2	(MSB)			IKE DESCRIPTOR LENGTH (000Ch)				
3								(LSB)
4	(MSB)			ALGORITHM IDENTIFIER				
•••								
7								(LSB)
8				Reserved				
•••								
11								

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 5.3.6.1. The IKE DESCRIPTOR LENGTH field shall be set as shown in table 71 for the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and the SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor.

The ALGORITHM IDENTIFIER field (see table 72) specifies Authentication payload authentication algorithm that applies to this SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor or SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor.

Table 72 — SA_AUTH_OUT and SA_AUTH_IN ALGORITHM IDENTIFIER field

Code	Description	Support	Reference	Authentication Reference [a]
00F9 0000h	SA_AUTH_NONE	Optional	this subclause	n/a
00F9 0001h	RSA Digital Signature with SHA-1 [b]	Optional	RFC 7296	4.1.3.3.3
00F9 0002h	Shared Key Message Integrity Code	Optional	RFC 7296 [c, d]	4.1.3.3.2
00F9 0009h	ECDSA with SHA-256 on the P-256 curve [b]	Optional	RFC 4754	4.1.3.3.3
00F9 000Bh	ECDSA with SHA-512 on the P-521 curve [b]	Optional	RFC 4754	4.1.3.3.3
00F9 00C9h to 00F9 00FFh	Vendor Specific	Optional		
0000 0000h to 0000 FFFFh	Restricted	Prohibited	IANA	
All others	Reserved			

[a] Description of how the algorithm shall be used to generate and verify the contents of the AUTHENTICATION DATA field of the Authentication Payload.

[b] Support for an RSA Digital Signature authentication algorithm does not mandate support for digital certificates.

[c] The 17 ASCII character non-terminated pre-shared key pad string 'Key Pad for IKEv2' specified by RFC 7296 is replaced by the 22 ASCII character non-terminated pre-shared key pad string 'Key Pad for IKEv2-SCSI'.

[d] The pre-shared key requirements used by this standard (see 4.1.3.3.2) apply in addition to those found in RFC 7296.

SA_AUTH_NONE specifies the omission of the IKEv2-SCSI Authentication step (see 4.1.3.7) as follows:

a) the presence of an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE is an SA Creation Capabilities payload (see 5.3.5.12) indicates that the device server is allowed to negotiate the omission of the IKEv2-SCSI Authentication step; and
b) the presence of an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE is an IKEv2-SCSI SA Cryptographic Algorithms payload (see 5.3.5.13) indicates the following based upon the command whose parameter data carries the payload:
   A) in the parameter list for a Key Exchange SECURITY PROTOCOL OUT command (see 4.1.3.6.2), SA_AUTH_NONE specifies that the application client is requesting that the IKEv2-SCSI Authentication step be skipped; and
   B) in the parameter data for a Key Exchange SECURITY PROTOCOL IN command (see 4.1.3.6.3), SA_AUTH_NONE indicates that the device server has agreed to skip the IKEv2-SCSI Authentication step.

If the device server indicates that skipping the Authentication step is allowed in its capabilities and the application client specifies that the Authentication step be skipped (see 4.1.3.3.4), then the IKEv2-SCSI Authentication step is skipped and the resulting SAs are not authenticated.

An SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE shall not appear in an IKEv2-SCSI SA Cryptographic Algorithms payload except as described in 4.1.3.3.4.

The Shared Key Message Integrity Code is based on a pre-shared key (see 4.1.3.3.2) that is associated with the identity in the Identification payload (see 5.3.5.4).

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 5.3.8.3 if the parameter list contains:

a) no SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptors;
b) no SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptors;
c) more than one SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor;
d) more than one SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor;
e) an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload (see 5.3.5.12) last returned by the device server to any application client;
f) an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload last returned by the device server to any application client;
g) an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 72;
h) an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 72;
i) an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE, and an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value other than SA_AUTH_NONE; or
j) an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE, and an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value other than SA_AUTH_NONE.

5.3.7 Errors in IKEv2-SCSI security protocol commands

For a single I_T_L nexus, the device server shall ensure that two or four IKEv2-SCSI CCS commands are processed in the order described in 4.1.3.1 as shown in table 73. For each SECURITY PROTOCOL OUT command shown in table 73, the parameter data shall not be processed unless table 73 specifies that the command is processed.

Table 73 — IKEv2-SCSI command ordering processing requirements on a single I_T_L nexus
(part 1 of 2)

	Time →				
IKEv2-SCSI SECURITY PROTOCOL OUT command or SECURITY PROTOCOL IN command received	**Before Key Exchange step SECURITY PROTOCOL OUT command completes with GOOD status**	**After a Key Exchange step SECURITY PROTOCOL OUT command completes with GOOD status**	**After a Key Exchange step SECURITY PROTOCOL IN command completes with GOOD status**	**After an Authentication step SECURITY PROTOCOL OUT command completes with GOOD status**	**After an Authentication step SECURITY PROTOCOL IN command completes with GOOD status**
Key Exchange step SECURITY PROTOCOL OUT command	Process the command as described in this standard	Do not process the command [a]	Do not process the command [a]	Do not process the command [a]	Same as before Key Exchange step SECURITY PROTOCOL OUT command
Key Exchange step SECURITY PROTOCOL IN command	No IKEv2-SCSI CCS exists [b]	Process the command as described in this standard	Repeat processing of the command [c]	Do not process the command [a]	Do not process the command [a]

[a] The command shall be terminated and the IKEv2-SCSI CCS shall be continued as follows:
a) the device server shall terminate the command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
b) the device server shall continue the IKEv2-SCSI CCS by preparing to receive another IKEv2-SCSI CCS SECURITY PROTOCOL OUT command.

[b] The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

[c] Processing of the SECURITY PROTOCOL IN commands in an IKEv2-SCSI CCS may be repeated. The device server should save the information necessary to repeat processing of these commands until the number of seconds specified in the IKEV2-SCSI PROTOCOL TIMEOUT field of the IKEv2-SCSI Timeout Values payload (see 5.3.5.15) have elapsed since the processing of the Authentication step SECURITY PROTOCOL OUT command that completed with GOOD status.

Table 73 — IKEv2-SCSI command ordering processing requirements on a single I_T_L nexus
(part 2 of 2)

IKEv2-SCSI SECURITY PROTOCOL OUT command or SECURITY PROTOCOL IN command received	Time → Before Key Exchange step SECURITY PROTOCOL OUT command completes with GOOD status	After a Key Exchange step SECURITY PROTOCOL OUT command completes with GOOD status	After a Key Exchange step SECURITY PROTOCOL IN command completes with GOOD status	After an Authentication step SECURITY PROTOCOL OUT command completes with GOOD status	After an Authentication step SECURITY PROTOCOL IN command completes with GOOD status
Authentication step SECURITY PROTOCOL OUT command	No IKEv2-SCSI CCS exists [b]	Do not process the command [a]	Process the command as described in this standard	Do not process the command [a]	Do not process the command [a]
Authentication step SECURITY PROTOCOL IN command		Do not process the command [a]	Do not process the command [a]	Process the command as described in this standard	Repeat processing of the command [c]
Command with an invalid field in the CDB	No IKEv2-SCSI CCS exists [b]	Do not process the command [a]	Do not process the command [a]	Do not process the command [a]	No IKEv2-SCSI CCS exists [b]

[a] The command shall be terminated and the IKEv2-SCSI CCS shall be continued as follows:
a) the device server shall terminate the command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
b) the device server shall continue the IKEv2-SCSI CCS by preparing to receive another IKEv2-SCSI CCS SECURITY PROTOCOL OUT command.

[b] The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

[c] Processing of the SECURITY PROTOCOL IN commands in an IKEv2-SCSI CCS may be repeated. The device server should save the information necessary to repeat processing of these commands until the number of seconds specified in the IKEV2-SCSI PROTOCOL TIMEOUT field of the IKEv2-SCSI Timeout Values payload (see 5.3.5.15) have elapsed since the processing of the Authentication step SECURITY PROTOCOL OUT command that completed with GOOD status.

The processing shown in table 73 shall be performed before parameter data error handling described in 5.3.8.

5.3.8 Errors in IKEv2-SCSI security protocol parameter data

5.3.8.1 Overview

Errors in the parameter data transferred to the device server by an IKEv2-SCSI SECURITY PROTOCOL OUT command are classified (see table 74) based on the ease with which they may be used to mount denial of service attacks against IKEv2-SCSI SA creation operations by an attacker that has not participated as the application client or device server in the Key Exchange step SECURITY PROTOCOL OUT command (see 4.1.3.6.2) that started the IKEv2-SCSI CCS.

Table 74 — IKEv2-SCSI parameter error categories

Denial of service attack potential	Applicable SECURITY PROTOCOL OUT commands	Error handling description	Reference
High [a]	Authentication step, and Delete operation	If the device sever is able to, the IKEv2-SCSI CCS state maintained for an I_T_L nexus is not changed	5.3.8.2
Low [b]	Key Exchange step, Authentication step, and Delete operation	The IKEv2-SCSI CCS state maintained for an I_T_L nexus is abandoned	5.3.8.3

[a] Attacks capable of causing significant harm by sending a malformed IKEv2-SCSI SECURITY PROTOCOL OUT command to the device server.

[b] Attacks that produce no significant harm, or collusive attacks (i.e., attacks that require knowledge of the IKEv2-SCSI CCS shared keys and participation in the IKEv2-SCSI CCS). Collusive attacks depend on collusion between the attacker and the application client (i.e., require the application client to act against its own best interests). The device server may abandon the IKEv2-SCSI CCS in response to such attacks.

5.3.8.2 Errors with high denial of service attack potential

Errors detected before or during the decryption and integrity checking of an Encrypted payload in an Authentication step SECURITY PROTOCOL OUT command or a Delete operation SECURITY PROTOCOL OUT command have a high potential for being a denial of service attack against one or more application clients (see table 74 in 5.3.8.1).

The device server shall respond to these SECURITY PROTOCOL OUT command errors as follows:

a) the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE REJECTED; and
b) the device server shall continue the IKEv2-SCSI CCS, if any, by preparing to receive an Authentication step SECURITY PROTOCOL OUT command.

If a specific field is the cause for returning the SA CREATION PARAMETER VALUE REJECTED additional sense code, then the SKSV bit may be set to one. If the SKSV bit is set to one, then the SENSE KEY SPECIFIC field shall be set as defined in SPC-5.

5.3.8.3 Errors with low denial of service attack potential

The errors that have low denial of service attack potential for IKEv2-SCSI SA creation (see table 74 in 5.3.8.1) are:

a) all errors detected for a Key Exchange step SECURITY PROTOCOL OUT command or its associated parameter data (e.g., errors detected in the IKEv2-SCSI header) that is attempting to establish a new IKEv2-SCSI CCS on an I_T_L nexus; and
b) all errors that are detected after the Encrypted payload has been successfully decrypted and integrity checked in an Authentication step SECURITY PROTOCOL OUT command (see 4.1.3.7.2) or a Delete operation SECURITY PROTOCOL OUT command (see 4.1.3.11).

The device server shall respond to these SECURITY PROTOCOL OUT command errors by terminating the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE INVALID. The state being maintained for an IKEv2-SCSI CCS on the I_T_L nexus on which the command was received, if any, shall be abandoned (see 4.1.3.10).

If a specific field is the cause for returning the SA CREATION PARAMETER VALUE INVALID additional sense code, then the SKSV bit shall be set to one and SENSE KEY SPECIFIC field shall be set as defined in SPC-5.

5.3.9 Translating IKEv2 errors

IKEv2 (see RFC 7296) defines an error reporting mechanism based on the Notify payload. This standard translates such error reports into the device server and application actions defined in this subclause.

If a device server is required by IKEv2 to report an error using a Notify payload, the device server shall translate the error into CHECK CONDITION status with the sense key and additional sense code as shown in table 75. The device server shall terminate the SECURITY PROTOCOL OUT command (see SPC-5) that transferred the parameter list in which the IKE requirements for one or more payloads (see 5.3.5) require use

of the Notify payload to report an error. The device server shall terminate the SECURITY PROTOCOL OUT command as described in this subclause. The device server shall not report the IKE errors described in this subclause by terminating a SECURITY PROTOCOL IN command (see SPC-5) with CHECK CONDITION status.

Table 75 — IKEv2 Notify payload error translations for IKEv2-SCSI

IKEv2 (see RFC 7296)		IKEv2-SCSI	
Error Type	**Description**	**Additional sense code**	**Sense key**
0000h	Reserved		
0001h	UNSUPPORTED_CRITICAL_PAYLOAD	SA CREATION PARAMETER NOT SUPPORTED	ILLEGAL REQUEST
0004h	INVALID_IKE_SPI	SA CREATION PARAMETER VALUE INVALID or SA CREATION PARAMETER VALUE REJECTED	
0005h	INVALID_MAJOR_VERSION		
0007h	INVALID_SYNTAX [a]		
0009h	INVALID_MESSAGE_ID		
000Bh	INVALID_SPI	SA CREATION PARAMETER VALUE INVALID [b]	
000Eh	NO_PROPOSAL_CHOSEN [c]	SA CREATION PARAMETER VALUE INVALID	
0011h	INVALID_KE_PAYLOAD [c]		
0018h	AUTHENTICATION_FAILED	AUTHENTICATION FAILED	ABORTED COMMAND
0022h to 0027h	See RFC 7296 [d]	n/a	n/a
2000h to 3FFFh	Vendor Specific		
All others	Restricted (see RFC 7296)		

[a] This sense key and one of the additional sense codes shown shall be returned for a syntax error within an Encrypted payload (see 5.3.5.11) regardless of conflicting IKEv2 requirements.

[b] SA CREATION PARAMETER VALUE INVALID shall be used for an invalid SAI in an IKEv2-SCSI SECURITY PROTOCOL IN or SECURITY PROTOCOL OUT. The additional sense code for an invalid SAI in all other commands is specified by the applicable usage type definition (see table 4 in 4.1.1.2).

[c] An application client recovers by restarting processing with the Device Capabilities step (see 4.1.3.5) to rediscover the device server's capabilities.

[d] These IKEv2 Error Types are used for features that are not supported by IKEv2-SCSI SA creation.

If an application client detects an IKEv2 error that RFC 7296 requires to be reported with a Notify payload, the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 4.1.3.10.

Annex A
(Informative)

Security goals and threat model

A.1 Introduction

In many cases, the security goals and threat model used for the Internet are applicable to SCSI commands. The Internet security goals and threat model found in RFC 3552 as they apply to SCSI are summarized in Annex B. Terms, concepts, and classes of security techniques that are defined in RFC 3552 are discussed based on their RFC 3552 definitions without modification in this standard.

The security goals and threat model described in Annex B are valid for all SCSI device types. Command standards may modify this model to handle threats appropriate to specific device types.

A.2 Security goals

The overall goals of security may be divided into the following categories:

a) communications security (i.e., protecting communications); and
b) system security (e.g., protecting systems from unauthorized usage, inappropriate usage, and denial of service).

These goals interact as a result of communications being carried out by systems, with access to those systems provided through communications channels. A common methodology is to secure the communications first and then provide secure access to systems over the secured communication channels.

Communication security is subdivided into the following primary areas of protection:

a) **confidentiality:** preventing unintended entities from seeing the data;
b) **cryptographic data integrity:** ensuring that the data that arrives is identical to the data that was sent; and
c) **peer entity authentication:** ensuring that the communicating endpoints are the intended peer entities.

Data origin authentication (i.e., ensuring that the received data was sent by the authenticated peer) is the combination of peer entity authentication and cryptographic data integrity.

Non Repudiation enhances data origin authentication with the ability to prove to a third party that the sender sent the data that the receiver received.

Cryptographic data integrity is called data integrity in RFC 3552. The term cryptographic is added in this standard to distinguish the class of integrity protection required to counter malicious attacks from the class of integrity protection required to deal with random data corruption (e.g., caused by cosmic rays or electrical noise). Mechanisms used to deal with random data corruption (e.g., parity bits and CRCs) have minimal value against malicious attacks that are able to modify integrity checks to conceal their modifications to the data. Cryptographic data integrity requires knowledge of a secret key in order to modify an integrity check without that modification being detectable. Systems should provide a high level of assurance that an attacker is unable to learn, guess, discover, or otherwise obtain the required secret key.

In addition to the primary areas, there is another area of control:

a) **authorization:** controlling what an entity is allowed to do. For communications security this is control of the entities with which an entity is allowed to communicate.

A form of authorization is access control (i.e., controlling what an entity is allowed to access).

A.3 Threat model

Most secured systems are vulnerable to an attacker equipped with sufficient resources, time, and skills. In order to make designing a security system practical, a threat model is defined to describe the capabilities that an attacker is assumed to be able to deploy (e.g., knowledge, computing capability, and ability to control the system).

The main purposes of a threat model are as follows:

a) to identify the threats of concern; and
b) to rule some threats explicitly out of scope.

Most security measures do not provide absolute assurance that an attack has not occurred. Rather, security measures raise the difficulty of accomplishing the attack to beyond the attacker's assumed capabilities and/or resources. Design of security measures that resist attackers with essentially unlimited capabilities (e.g., certain nation-states) is outside the scope of this standard. Security measures that are susceptible to a level of capability available to some attackers may still be useful for deterring attackers who lack that level of capability, especially when combined with non-technical security measures such as physical access controls.

The computational capability of an attacker is treated as a variable because that capability is inherently a moving target as a result of more powerful processors. The computational capability of an attacker influences design aspects (e.g., key length). Well designed security systems are agile in that they are able to operate not only with different key lengths, but also with different cryptographic algorithms.

The Internet threat model described in RFC 3552 is generally applicable to SCSI, and is specifically applicable if Internet Protocols are used by the SCSI transport (e.g., iSCSI, Fibre Channel via FCIP, or Fibre Channel via iFCP). The basic assumptions of the Internet threat model are:

a) end systems engaging in communication are not under the control of the attacker; and
b) the attacker is able to read any communicated data (e.g., data in an IU) and undetectably remove, change, or inject forged IUs, including injection of IUs that appear to be from a known and/or trusted system.

Communications security designs are based on an additional assumption that secrets (e.g., keys) used to secure the communications are protected so that an attacker is unable to learn, guess, discover, or otherwise obtain them. A consequence of this assumption is that attacks against secured communications are assumed to begin without with advance knowledge of the secrets used to secure the communications.

A.4 Types of attacks

The following types of attacks are considered:

a) passive attacks (i.e., attacks that only require reading IUs); and
b) active attacks (i.e., attacks that require the attacker to change communication and/or engage in communication).

More information on attack types is available in RFC 3552.

Simple passive attacks involve reading communicated data that the attacker was not intended to see (e.g., password, credit card number). More complex passive attacks involve post-processing the communicated data (e.g., checking a challenge-response pair against a dictionary to see if a common word was used as a password).

There are a wide variety of active attacks (e.g., spoofing, replay, insertion, deletion, known plaintext, and modification of communications). Man-in-the-middle attacks are a class of active attacks that involve the attacker inserting itself in the middle of communication, enabling the attacker to intercept all communications without the knowledge of the communicating parties for various purposes (e.g., insertion, deletion, replay, modification and/or inspection via decryption of the communications).

A.5 SCSI security considerations

The application of communication security techniques (see RFC 3552) is defined by command standards. This subclause describes specific design considerations in applying the threat model (see A.3) to all SCSI device types.

SCSI environments tend not to be fully connected (i.e., there are restrictions on the SCSI device servers with which a SCSI application client is able to communicate) due to the following mechanisms:

a) physical and logical connectivity restrictions (e.g., in SCSI to SCSI gateways across different transports);
b) LUN mapping and masking; and
c) transport zoning.

The resulting connectivity is more limited than the Internet security assumption that an off-path attacker is able to transmit to an arbitrary victim (see RFC 3552).

SCSI security designs are also influenced by SCSI being a client-server distributed service model (see SAM-5) that is realized over a number of different SCSI transport protocols and interconnects.

Security functionality may be defined as part of a command set or at the SCSI transport level. Some SCSI transport protocols (e.g., Fibre Channel and iSCSI) define security functionality that provides confidentiality, cryptographic integrity, and peer entity authentication for all communicated data. However, there are situations in which some or all of those mechanisms are not used and there are SCSI communications whose scope spans more than one SCSI transport protocol (e.g., via a gateway between iSCSI and FCP). Security that is defined by a command set is appropriate for such situations.

Annex B
(Informative)

Variations between this standard and equivalent security protocols

B.1 IKEv2 protocol details and variations for IKEv2-SCSI

The IKEv2 protocol details and variations defined in RFC 7296 apply to IKEv2-SCSI (i.e., this standard) as follows:

a) any SECURITY PROTOCOL OUT command with a transfer length of up to 16 384 bytes is not terminated with an error due to the number of bytes transferred;
b) the timeout and retransmission mechanisms defined in RFC 7296 are not used by this standard, instead a new payload is defined to transfer appropriate timeout values to the device server;
c) each SCSI command used by this standard completes by conveying a status from the device server to the application client;
d) the IKEv2 header EXCHANGE TYPE field is reserved in this standard as a result of equivalent information being transferred in the SECURITY PROTOCOL OUT command and SECURITY PROTOCOL IN command CDBs;
e) the IKEv2 header VERSION bit is reserved in this standard;
f) this standard uses the Notify payload (see 5.3.5.9) only to provide initial contact information;
g) this standard uses the pseudo-random functions defined by RFC 7296;
h) the key derivation functions defined and used by this standard (see 4.1.2) are equivalent to the PRF+ found in RFC 7296;
i) the SA creation transactions defined by this standard are not overlapped. If an application client attempts to start a second SA creation transaction before the first is completed, then the offending command is terminated as described in 4.1.3.1, but this does not affect the SA creation transaction that is already in progress;
j) the NO_PROPOSAL_CHOSEN notify error type and INVALID_KE_PAYLOAD notify error type are replaced by the SA CREATION PARAMETER VALUE INVALID additional sense code (see 5.3.9) because IKEv2-SCSI has a different negotiation structure. As defined in RFC 7296, an IKEv2 initiator offers one or more proposals to a responder without knowing what is acceptable to the responder, and chooses a DH group without knowing whether that DH group is acceptable to the responder. These two notify error types allow the responder to inform the initiator that one or more of its choices are not acceptable. In contrast, an IKEv2-SCSI application client obtains the device server capabilities in the Device Capabilities step (see 4.1.3.5) and selects algorithms from them in the Key Exchange step (see 4.1.3.6). An error only occurs if the application client has made an invalid selection, the response to which includes the SA CREATION PARAMETER VALUE INVALID additional sense code;
k) IKEv2 version numbers (see RFC 7296) are used by this standard (see 5.3.4), but the ability to respond to an unsupported version number with the highest version number to be used is not supported, and this standard does not include checks for version downgrade attacks;
l) IKEv2 cookies (see RFC 7296) are not used by this standard;
m) IKEv2 cryptographic algorithm negotiation (see RFC 7296) is replaced by the Device Server Capabilities step (see 4.1.3.5) and the Key Exchange step (see 4.1.3.6) (i.e., the IKEv2 proposal construct is not used by this standard);
n) in this standard an SA is rekeyed by replacing it with a new SA:
    A) CHILD_SAs are not used by this standard;
    B) the RFC 7296 discussion of CHILD_SAs does not apply to this standard;
    C) coexistence of the original SA and the new SA is achieved for rekeying purposes by restricting the device server's ability to delete SAs to the following cases:
        a) expiration of a timeout (see 5.3.5.15);
        b) processing of an IKEv2-SCSI Delete function (see 4.1.3.11); and
        c) responding to an initial contact notification (see 5.3.5.9);

and
   D) IKEv2 does not support rekeying notification for IKE_SAs, therefore this standard does not support rekeying notification;
o) the choice of authentication methods for both transfer directions is negotiated using the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor (see 5.3.6.6) during the Key Exchange step (see 4.1.3.6);
p) the usage of Certificate Encodings in the Certificate payload (see 5.3.5.5) and Certificate Request payload (see 5.3.5.6) are constrained as follows:
   A) in accordance with the recommendations in RFC 7296, Certificate Encoding values 01h to 03h and 05h to 0Ah are prohibited;
   B) this standard forbids the use of URL-based Certificate Encodings (i.e., Certificate Encodings values 0Ch and 0Dh); and
   C) certificate Encoding values that RFC 7296 defines as vendor specific are reserved in this standard;
q) deleting an SA requires knowing the SAIs (i.e., SPIs) in both directions and including both SAIs in the Delete payload. The RFC 7296 description of the Delete payload is vague enough to allow this. The requirement is consistent with the SCSI model for SAs;
r) the Vendor ID payload is not used by this standard;
s) Traffic Selectors (see RFC 7296) are not used by this standard;
t) the requirements in RFC 7296 on nonces are to be followed for the random nonces defined by this standard;
u) the RFC 7296 requirements on address and port agility are specific to the user datagram protocol and the IP protocol and do not apply to this standard;
v) keys for the Authentication step are generated as defined in RFC 7296;
w) this standard uses a slightly modified version of the authentication calculations in RFC 7296 (see 5.3.5.7);
x) the RFC 7296 sections that describe the following features are not used by this standard:
   A) extensible authentication protocol methods;
   B) generating keying Material for CHILD_SAs;
   C) rekeying an IKE SA using CREATE_CHILD_SA;
   D) requesting an internal address;
   E) requesting the peer's version;
   F) IPComp;
   G) NAT traversal; and
   H) explicit congestion notification;
y) IKEv2 Error Handling (see RFC 7296) is replaced by the use of CHECK CONDITION status and sense data by this standard. See 5.3.9 for details of how errors reported in the Notify payload are translated to sense data;
z) IETF standards omit the Integrity transform instead of using AUTH_COMBINED;
aa) the command-response architecture of SCSI is susceptible to denial of service attacks, and no protection against such attacks is defined in this standard. Protection against denial of service attacks against the application client is described in 5.3.8;
ab) IKEv2-SCSI requires Encrypted Payloads be padded to at least the 4 byte minimum alignment required by ESP-SCSI, whereas IKEv2 imposes no such requirement;
ac) the critical (CRIT) bit is set to one in all IKEv2-SCSI payloads defined in this standard and these payloads are required to be recognized by all IKEv2-SCSI implementations. RFC 7296 sets the Critical (C) bit to zero in all IKEv2 payloads defined in RFC 7296, and requires that all IKEv2 implementations recognize all payloads defined in RFC 7296; and
ad) use of the Identification payloads is required by IKEv2-SCSI, whereas IKEv2 allows the Identification payloads to be omitted.

Where this standard uses IKE payload names (see 5.3.4) RFC 7296 uses the shorthand notation shown in table B.1.

Table B.1 — IKE payload names shorthand

IKE payload name in this standard [a]	RFC 7296 shorthand [b]
Security Association	SAi or SAr
Key Exchange	KEi or KEr
Identification – Application Client	IDi
Identification – Device Server	IDr
Certificate	CERTi or CERTr
Certificate Request	CERTREQi
Authentication	AUTHi or AUTHr
Nonce	NONCEi or NONCEr
Notify	N-ICi or N-ICr
Delete	Di
Vendor ID	Vi or Vr
Traffic Selector – Application Client	TSi
Traffic Selector – Device Server	TSr
Encrypted	Ei or Er
Configuration	CPi or CPr
Extensible Authentication	EAPi or EAPr

[a] To facilitate future enhancements, all IKE payloads are listed in this table, but not all entries in this table are used in this standard.

[b] In RFC 7296 the lowercase i indicates initiator and r indicates responder. In this standard, the initiator is the application client and all such IKE payloads (e.g., KEi) appear in a SECURITY PROTOCOL OUT parameter list. The responder is always the device server in this standard and all such IKE payloads (e.g., AUTHr) appear in SECURITY PROTOCOL IN parameter data.

B.2 ESP protocol details and variations for ESP-SCSI

The IKEv2 protocol details and variations defined in RFC 4303 apply to ESP-SCSI (i.e., this standard) as follows:

a) this standard requires an integrity check value (e.g., INTEGRITY CHECK VALUE fields), whereas ESP allows support of confidentiality-only;
b) this standard does not support traffic flow confidentiality;
c) this standard does not support the TCP/IP (see bibliography) aspects of ESP (e.g., IP addresses, multicast);
d) this standard requires anti-replay detection using the sequence number, whereas ESP makes this optional;
e) this standard does not support the Next Header field, but does reserve space for it in the MUST BE ZERO field (see table 14 in 4.1.5.3);
f) this standard requires verification of the padding bytes, when possible;
g) there is no provision in this standard for generating "dummy packets"; and
h) this standard does not support out-of-order parameter data.

Bibliography

ANSI INCITS 519-2014, *Serial Attached SCSI - 3 (SAS-3)*

ISO/IEC 14776-321, *SCSI Block Commands (SBC) [ANSI INCITS 306-1998-R2008]*

NOTE 12 - SBC is the only published standard that defines write once devices and optical memory devices.

ANSI INCITS 494-2012, *Automation/Drive Interface - Commands - 3 (ADC-3)*

ANSI INCITS 309-1998, *Serial Storage Architecture SCSI-3 Protocol (SSA-S3P)*

ANSI INCITS 441-2008, *Automation/Drive Interface - Transport Protocol - 2 (ADT-2)*

ANSI INCITS 496-2012, *Fibre Channel Security Protocols-2 (FC-SP-2)*

ISO 80000-1:2009, *Quantities and units – Part 1: General*

ISO 80000-2:2009, *Quantities and units – Part 2: Mathematical signs and symbols to be used in the natural sciences and technology*

IEC 80000-13:2008, *Quantities and units – Part 13: Information science and technology*

RFC 791, *Internet Protocol - DARPA Internet Program - Protocol Specification*

RFC 1035, *Domain Names - Implementation and Specification*

RFC 1591, *Domain Name System Structure and Delegation*

RFC 2753, *A Framework for Policy-based Admission Control*

RFC 3447, *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*

RFC 3552, *Guidelines for Writing RFC Text on Security Considerations*

RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*

RFC 6874, *Representing IPv6 Zone Identifiers in Address Literals and Uniform Resource Identifiers*

RFC 7320, *URI Design and Ownership*

RFC 7414, *A Roadmap for Transmission Control Protocol (TCP) Specification Documents*

NOTE 13 - Copies of the IETF RFCs may be obtained at http://www.ietf.org/.